PAPER

A Hybrid Genetic Service Mining Method Based on Trace Clustering Population

Yahui TANG[†], **Tong LI**[††a)], **Rui ZHU**[†††], ***Nonmembers***, **Cong LIU**[††††], ***Member***, ***and*** **Shuaipeng ZHANG**[††††], ***Nonmember***

SUMMARY Service mining aims to use process mining for the analysis of services, making it possible to discover, analyze, and improve service processes. In the context of Web services, the recording of all kinds of events related to activities is possible, which can be used to extract new information of service processes. However, the distributed nature of the services tends to generate large-scale service event logs, which complicates the discovery and analysis of service processes. To solve this problem, this research focus on the existing large-scale service event logs, a hybrid genetic service mining based on a trace clustering population method (HGSM) is proposed. By using trace clustering, the complex service system is divided into multiple functionally independent components, thereby simplifying the mining environment; And HGSM improves the mining efficiency of the genetic mining algorithm from the aspects of initial population quality improvement and genetic operation improvement, makes it better handle large service event logs. Experimental results demonstrate that compare with existing state-of-the-art mining methods, HGSM has better characteristics to handle large service event logs, in terms of both the mining efficiency and model quality.

key words: *service mining, process mining, trace clustering, genetic process mining algorithm*

1. Introduction

Service mining is defined as the method of using process mining for the analysis of services [1]. The starting point for process mining is an event log. By exploiting the data recorded during the operation of service-oriented systems, one can discover behavioral models to describe the actual operation of the service. The discovered behavioral models provide extensive insights into the real usage of the service, thereby enabling new forms of model-based testing and improvements.

The Web service is, itself, a software system [2]. Service event logs record the execution information of the software. However, during the execution of software systems, a considerable number of crashes and exceptions may occur, and the distributed nature of the services complicates the design and analysis of service-oriented systems that support end-to-end business processes. These lead to large-scale service event logs exhibiting a complicated structure and represent a major challenge in this field.

Mining algorithms aim to discover models from event logs. The genetic process mining algorithm (GPM) [3] is quality-guided, makes it easier to generate high-quality process models, as compared to other mining algorithms. However, the algorithm is less efficient for large-scale service event logs.

Based on the above issues, this research focus on processing the existing large-scale service event logs, proposes a hybrid genetic service mining method based on trace clustering population (HGSM). HGSM simplifies the mining environment by clustering traces in event logs. Meanwhile, HGSM improves the mining efficiency of the genetic mining algorithm to better handle large service event logs, which can mine high-quality service process models efficiently. The main contributions are as follows:

1. Use the trace clustering method to divide the complex software system into multiple functionally independent components, and them using the inductive miner (IM) [4] algorithm for pre-mining to prepare a high-quality initial population for GPM, which can reduce the number of iterations to reach the high-quality model, and making it easier to generate process models with high-quality;
2. Improvement of the mutation operation of the genetic operation, by changing the mutation operation from random to directed, thereby speeding up the population optimization;
3. An overall quality function is proposed to balance four quality dimensions of the model, and makes the model generated by the algorithm have higher overall quality.

2. Related Work

The autonomous nature of the services and the fact that they are loosely coupled make it important to monitor and analyze their behavior [5]. In the context of Web services, the recording of all kinds of events related to activities is possible, which can be used to extract new information of service processes [6]. Process mining aims to discover, monitor, and improve real processes by extracting knowledge from the event logs [7], which makes it possible to analyze service processes based on events recorded in the context of Web services.

Manuscript received September 21, 2021.
Manuscript revised December 7, 2021.
Manuscript publicized April 28, 2022.

[†]The author is with School of Information, Yunnan University, Kunming, 650500, China.
[††]The author is with School of Big Data, Yunnan Agricultural University, Kunming, 650201, China.
[†††]The author is with School of Software, Yunnan University, Kunming, 650500, China.
[††††]The authors are with School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China.
a) E-mail: tli@ynu.edu.cn

DOI: 10.1587/transinf.2021EDP7190

In the field of service mining, Taibi and Systä [8] proposed a decomposition method based on process mining, which could identify the micro-services, and separate them from the monolithic systems. Pérez-Castillo et al. [9] proposed a refactoring technique, which is specially developed for business process models, obtained by process mining, and other reverse engineering techniques. van der Aalst and Verbeek [10] employed a web sphere case to present the application of process mining in services.

Traditional process mining algorithms exhibit problems when dealing with large-scale service processes, and generate spaghetti-like process models that are hard to analyze [11]. An approach to overcome this is to cluster the traces so that each resulting cluster corresponds to a cohesive and similar set of process instances that can be fully represented by a model. Song et al. [12] cluster traces based on the event log profile, which can be applied to a flexible environment to improve the mining results. Bose and Aalst [13] cluster traces based on the universal edit distance. Hompes et al. [14] proposed a feature set clustering method based on conservative patterns, by selecting a specific clustering dimension that can be obtained by analyzing the event log, making it possible to obtain better clustering results.

The alignment technology can compare the behavior of the process model with the behavior recorded in the event log and discover their similarities and deviations [15]. However, the existing alignment techniques are limited to finding the differences, and are usually used to evaluate the quality of the model, the repair of the model can only be completed by the users [15].

GPM can generate higher-quality models compared with other mining methods. However, the algorithm is less efficient. One reason is that the initial population quality of GPM is poor due to its random generation, the other is mutation operation in GPM is performed in a random manner, which makes it difficult to optimize the model quality. Liqin et al. [16] present a genetic mining algorithm based on case classification, which uses three existing mining algorithms to prepare the initial population to improve the mining results of the genetic algorithm. However, these mining methods can not ensure the soundness [4] of the population model, and too many individuals in the population will increase the processing time of the genetic mining algorithm. And the mutation operation of the proposed algorithm is performed randomly.

The proposed method uses IM algorithm to prepare a high-quality and soundness initial population, meanwhile changing the mutation operation from random to directed by using the information of alignment. These factors can accelerate the convergence speed of the genetic mining algorithm.

3. Methods

Figure 1 shows the architecture of HGSM. HGSM input event log, output process model. As Fig. 1 shows, Part 1 of the HGSM prepares the high-quality initial population for the genetic algorithm. To be specific, the trace of the event log is clustered by the Markov cluster algorithm (MCL) [14] to group it into N sub-logs, each of which is a group of similar traces on the selected attribute of trace. And the IM algorithm discovers N process models from N sub-logs. In part 2, as the initial population, N models are optimized and synthesized by genetic algorithm. Finally, a model with the best quality is obtained. HGSM has been implemented in the process mining toolkit ProM [16], and can be executed continuously.

3.1 High-Quality Initial Population Preparation

To divide the complex service process into loosely-coupled components and simplify the mining environment, we divide the event log based on MCL [14]. The MCL algorithm is a trace clustering method that inputs an event log, outputs sub-logs, each consisting of a similar set of traces. Firstly, for the analysis of an event log, an attribute of the trace is selected as the clustering dimension, which determines the clustering results and the number of clusters.

Almost all trace attributes can be used as clustering dimensions. One can select the clustering dimension according to the mining target. i.e., the components of a software, the execution time of a trace, the type of payment, etc. And the selection of the mining target may come from the prior knowledge from domain experts. If there is no mining target, different clustering dimensions can be used to cluster separately, and then the selection of clustering dimensions can be determined based on the mining results. Secondly, the algorithm uses the cosine similarity to create the trace similarity matrix, and each case in the event log is mapped to a multi-dimensional vector. Similarly, the attributes of

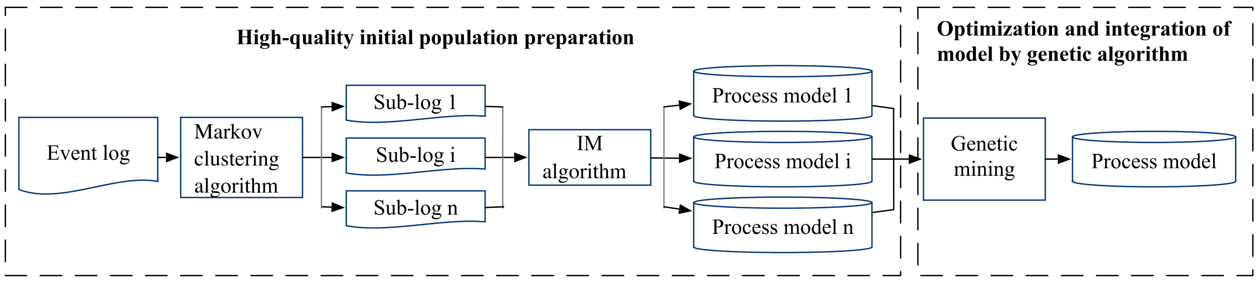

High-quality initial population preparation
Optimization and integration of model by genetic algorithm
Event log
Markov clustering algorithm
Sub-log 1
Sub-log i
Sub-log n
IM algorithm
Process model 1
Process model i
Process model n
Genetic mining
Process model

Fig. 1 Architecture of HGSM.

the case are mapped to the different dimensions of the vector. Finally, the clustering is based on the similarity between the vectors. In practice, a similar set of traces often corresponds to a particular process containing coherent behavior. It is easy to get better mining results by mining such a group of traces.

Definition 1 (**Event, Activity, Attribute** [14]) Let ε be the event universe, i.e., the set of all possible event identifiers. Events may be characterized by various attributes, e.g., an event may correspond to an activity, or be executed by a particular person. Let N be a set of attribute names. For any event $e \in \varepsilon$ and attribute name $n \in N$, $\#n(e)$ is the value of attribute n for event e. Let A be the activity universe, i.e., the set of all possible activity identifiers. $\#\text{act}(e) \in A$ is the activity related to e.

Definition 2 (**Case, Trace, Event log** [14]) Let C be the case universe, i.e., the set of all possible case identifiers. Cases, such as events, have attributes. For any case $c \in C$ and attribute name $n \in N$, $\#n(c)$ is the value of attribute n for case c. Each case has a mandatory attribute called "trace": $\#\text{trace}(c) \in C^*$. $\hat{c} = \text{trace}(c)$ is the shorthand notation referring to the trace of a case. A trace is a finite sequence of events $\sigma \in \varepsilon^*$, such that each event appears only once, i.e., for $1 \leq i \leq j \leq |\sigma|$: $\sigma_i \neq \sigma_j$. An event log is a set of cases $L \subseteq C$ such that each event appears at most once in the entire log, i.e., for any $c_1, c_2 \in L$ such that $c_1 \neq c_2$: $\partial_{set(c_1)} \cap \partial_{set(c_2)} = \emptyset$.

Definition 3 (Trace Similarity Matrix [14]) Let $L \subseteq C$ be an event log. $S(L) = (L \times L) \to [0.0, 1.0]$ denotes the set of all possible trace similarity matrices over L. For cases $l, l' \in L$, and a trace similarity matrix $S \in (L)$, $S(l, l')$ denotes the similarity between l and l'.

Definition 4 (Trace clustering [14]**):** Given an event log L, a trace clustering over L is a partition over a (possibly proper) subset of the traces in L.

Next, sub-models are discovered by the IM algorithm. The IM algorithm is a discovery approach to generate a process model for a given log. IM can construct high-quality process models form with more than 80% replay fitness in polynomial time. Meanwhile, the IM algorithm can ensure that all the models mined are sound [4]. An unsound model may contain livelock, deadlock, or other anomalies. In each iteration of the genetic algorithm, the quality of the models needs to be measured and improved, and if the model is unsound, additional work is needed to repair the model. In summary, the IM algorithm is applicable to prepare a better initial population.

The approach works recursively with the divide and conquer strategy: split log, construct part of the process tree, and proceed with handling split parts of log separately [4]. To be specific, the input of the algorithm is the event log L, and the output is the discovered process model N. The IM algorithm uses a divide-and-conquer strategy to split the event log L into m sub-logs, simplifying the problem to find m sub-process models for the m sub-logs. The steps are as follows: first, select appropriate slicing operators (sequential, parallel, loop, and exclusive choice) for log L, and divide the activities in log L into disjoint sets through slicing; These sets then correspond to the sub-logs $L_1, L_2, \dots, L_m$ resulting from the splitting of log L; next, through the above steps, the sub-logs $L_1, L_2, \dots, L_m$ are recursively mined until they contain only one activity. Finally, the slicing operator records how activities are grouped together to form a process tree language, which can be used to return a complete process tree model.

3.2 Optimization and Integration of Model by Genetic algorithm

1. Representation of process model

Only a small fraction of all possible models represented in traditional languages such as BPMN, Petri nets, UML activity diagrams are sound [7]. HGSM is a quality-guided mining algorithm that requires multiple measurements of model quality. For this, such unsound constructs should be avoided. Therefore, we use the process tree as the representation of the model, since all possible process trees are sound [17]. This avoids the extra work of model repair and improves the performance of the genetic algorithm.

Definition 5 (**Process tree** [4]**):** Process tree is defined recursively, let $\bigotimes$ be a finite set of operators, A is a set of activities, tau $\notin$ A is the silent activity.

$a \in (A \cup \{tau\})$ is a process tree;

Let $P_1, \dots P_n$ $(n > 0)$ be process trees and $\otimes \in \bigotimes$ be a process tree operator, then $\otimes(P_1, \dots P_n)$ is a process tree.

The following are the standard operators for the process tree:

$\times$ is the exclusive choice between one of the subtrees;

$\to$ is the sequential execution of all subtrees;

$\|$ indicates the parallel execution of all subtrees;

$\circlearrowleft$ is a loop operator with loop body P_1 and optional loop paths $P_2, \dots P_n$ $(n \geq 2)$.

2. Fitness function

Fitness function is the optimization goal of genetic algorithm. Four quality dimensions have been proposed for model evaluation: replay fitness, precision, generalization, and simplicity [17]. And the quantitative methods for the four quality dimensions are as follows:

1) Replay fitness:

Replay fitness quantifies the degree to which the model can align with the behavior recorded in the log [18]. This study uses an alignment-based fitness computation defined in [18], it aligns the event log with the process model to observe the inconsistency between them, which is calculated as Eq. (1). For the inconsistency, we can change the activities in the model to make them consistent, and the effort required to change them is the "cost". *Min cost* is the minimal cost of aligning the event log to the model with no synchronous moves, and *Actual cost* is the actual cost for the aligning model and event log. The fitness value is in the range [0, 1].

$$\textit{Replay fitness} = \frac{\textit{Min cost}}{\textit{Actual cost}} \tag{1}$$

2) Simplicity:

Simplicity indicates the complexity of the model, simplicity is measured according to [17] by comparing the size of the model to the number of activities in the log. The simplicity value defined in [17] is given by Eq. (2). *#duplicate activities* is the number of repeat times of activity in the process model. The missing activity is the activity that is recorded in the event log but not included in the model. These numbers are summed and normalized by the total number of nodes in the model and the activities in the event log.

$$Simplicity = \frac{\#duplicate\ activities + \#missing\ activities}{\#nodes\ in\ model + \#total\ activities} \tag{2}$$

3) Precision:

Precision illustrates underfitting is avoided [19], which indicates the behaviors allowed by the process model but not appearing in the event log. The precision measurement method refers to [17], which compares the state space executed by the process tree when replaying the log, and calculates the precision according to the escaping edge, that is, the decision appears in the model, but does not appear in the log. If there is no escape edge, the precision is perfect. We obtain the part of the state space used from information provided by the replay fitness, where we ignore events that are in the log, but do not correspond to an activity according to the alignment. we calculate the precision as follows:

$$Precision = 1 - \frac{\sum_{visited\ markings} \#visits * \frac{\#outgoing\ edges - \#used\ edges}{\#outgoing\ edges}}{\#total\ marking\ visits\ over\ all\ markings} \tag{3}$$

4) Generalization:

Generalization signifies avoiding overfitting [20]. Generalization estimates how well the process model describes the (unknown) system, and not only the event log. If all parts of the process model are frequently used, the process model is likely to be generic. However, if some parts of the process model are rarely used, chances are high that the system allows for more behavior. Therefore we base the generalization measure on how often parts of the process model have been used while replaying the event log. For this, we use the alignment provided by the replay fitness. If a node is visited more often then we are more certain that its behavior is (in)correct. If some parts of the tree are very infrequently visited, generalization is bad. Generalization is calculated as follows:

$$Generalization = 1 - \frac{\sum_{nodes}\left(\sqrt{\#executions}\right)^{-1}}{\#nodes\ in\ tree} \tag{4}$$

Although there are quantitative methods for the quality dimensions, they tend to compete with each other, and it is difficult to achieve a balance between them. Meanwhile, state-of-the-art methods can only take into account a part of the quality dimension [7] e.g., the model mined by the region-based mining algorithm [21] has a good replay fitness and precision, but poor generalization and simplicity. However, low values in any quality dimension can affect the mining results [17]. To balance these dimensions, we proposed the overall quality function (OQ), which is the non-weighted sum of the four dimensions given by (5). HGSM uses OQ function as the fitness function in the genetic algorithm to guide the mining in the process of the iterative mining model.

$$OQ = (Fr + Pe + Gn + Sm)/4 \tag{5}$$

Fr, Pe, Gn, and Sm are quantitative measures of the model in four dimensions, which is given by Eqs. (1)–(4). Their values are between 0 and 1.

3. Mutation operation based on alignment log

The genetic operations aim to improve the quality of the process tree in the population. They directly affect the process tree. To obtain good mining results, the population should have sufficient diversity and high quality. According to the proportion of the elite selection, multiple process trees having high OQ values are retained to the next generation, while the process trees in the rest of the population are improved by the genetic operations. There are three kinds of genetic operations: replacement, crossover, and mutation [3]. The replacement operation selects some process trees with the worst quality in the population according to a certain proportion, and then uses randomly generated process trees, to replace this part of the process trees; the crossover operation exchanges two randomly selected sub-trees, between the two process trees. Crossover and replacement are used to increase the population diversity and are unable to improve the quality of the process tree. The mutation is the only genetic operation that can improve the quality of the process trees, it directly operates on the nodes of the process tree.

To accelerate the convergence speed of the genetic mining algorithm, it is necessary to improve the quality of process trees in the population efficiently. However, in the genetic mining algorithm, the mutation is conducted randomly, i.e., randomly selecting the nodes and changing them, such a mutation is difficult to improve the quality of the process trees. Therefore, we improved the mutation operation.

The alignment technology can compare the behavior of the process model with that recorded in the event log and discover their similarities and deviations [15]. The fewer deviations between model and log, the higher the quality of the model. The deviation occurs when an activity only appears in the log or model, and this deviation can be called a move on model or a move on log [15]. The ">>" identifier is introduced to represent the deviation. However, the existing alignment techniques are limited to finding the deviations to measure the replay fitness of the model, the repair of the model can only be completed by the users [15].

This study uses the information obtained by aligning the log to guide the mutation operation when calculating the replay fitness of the process tree. The higher the quality of

Log	a	>>	c
Model	a	b	c
Log	a	>>	c
Model	a	b	c

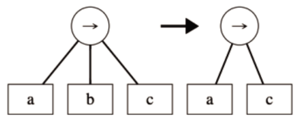


Fig. 2 b move on the log, remove it from the model.

Log	a	>>	c
Model	a	b	c
Log	a	b	c
Model	a	b	c

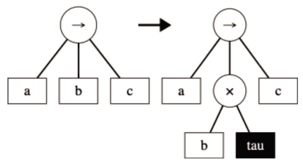


Fig. 3 b sometimes moves on the log, adds a tau activity to make b can skip.

Log	a	b	c
Model	a	>>	c
Log	b	a	c
Model	>>	a	c

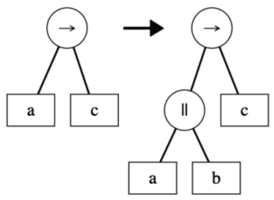


Fig. 4 The order of a and b is indefinite, add a brother node b under the parallel operation node for a.

Log	a	b	c
Model	a	>>	c
Log	a	b	c
Model	a	>>	c

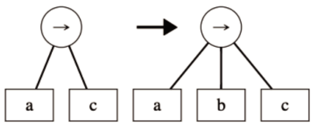


Fig. 5 b move on the model, add it to the model.

Log	a	c
Model	a	c
Log	b	c
Model	a	c

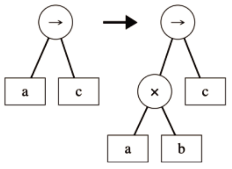


Fig. 6 If only a or b move on log, add the exclusive choice node as the parent node of them

Log	a	b	c	>>
Model	a	>>	c	b
Log	b	a	c	>>
Model	>>	a	c	b

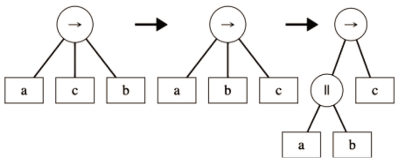


Fig. 7 Select an operator node to change the process tree.

the process tree in the population, the faster the algorithm converges [3]. By changing the mutation operation from random to directed, the model can get closer to the event log faster, thereby accelerating the quality improvement of the process tree, and ultimately speeding up the convergence speed of the genetic algorithm.

Definition 6 (Alignment, move [15]) Let $l = a_1, a_2, a_3 \ldots a_m$ be a trace over event log ε, let $t = s_1, s_2, s_3, \ldots s_m$ be a set of actually occurring activity sequences in process model N, an alignment from ε to N is a sequence $\alpha = (a_1, s_1)(a_2, s_2) \ldots (a_k, s_k)$, where:

The first line of alignment α: $(a_1, a_2, a_3 \ldots a_m)$ is a trace in the event log;

The second line of alignment α: $(s_1, s_2, s_3, \ldots s_m)$ is a set of activity occurrence sequences in process model N;

For all the i = 1, 2, ... k, $l(s_i) = a_i$, iff $l(s_i) \neq \text{tau} \wedge a_i \neq >> l \wedge s_i \neq \text{tau}$;

A move is a pair $(a, s) \in (l \cup \{>>\}) \times (t \cup \{>>\}) \setminus \{(>>, >>)\}$, such that:

(a_i, s_i) is a move on model, iff $a_i = >> \wedge s_i \neq >>$;

(a_i, s_i) is a move on log, iff $a_i \neq >> \wedge s_i = >>$;

(a_i, s_i) is a synchronous move, iff $a_i \neq >> \wedge s_i \neq >>$.

Mutation operation directly operates on nodes in the model, including adding nodes, deleting nodes, and changing nodes. And Figs. 2–7 are instances of guided mutation. The "Log" line in the figure means the sequence of activities recorded in the event log, and the "Model" line in the table means the sequence of activities reflected in the model. Process tree to the left of the arrow means the original process tree, and the right is the process tree processed by mutation operation. As shown in Fig. 2, if the selected leaf node is always skipped during the alignment (that is, the activity is only moved on the model), then the leaf will be removed, and if the selected leaf node is sometimes skipped, namely b activity, exclusive choice node and tau activity will be added to the model, which makes the node can be skipped, as shown in Fig. 3; If the order of the selected node and the node to be added is inconsistent, then a parallel parent node is added for them, as shown in Fig. 4; If the selected node always appears in the log, but not in the model, the corresponding node will be added in the model, i.e., b activity in Fig. 5; If there is only one of activities appears in the trace,

Algorithm1: HGSM for genetic algorithm

```
Input : Event log: L; Number of iterations N; α=0.25;
Process tree set: ProcessTrees={t1,t2,…,tN};
Output : A process tree: PT; The OQ value of PT;
1:  foreach (i: [1,2,…,N])
2:    foreach (tree: [t1,t2,…,tN] )
3:         oquality(tree);    // Calculate the OQ value of each tree.
4:         sort(tree);         // Sort in descending order according to the OQ value.
5:    end foreach
6:    foreach (tree: [t1,t2,…,tN] )
7:         eliteSelectionSet={tj | 1≤j<αN}; // According to OQ values, the top 25% of the process trees are kept to the next generation.
8:         mutationSet={mutation(tj) | αN≤j<2αN }// Mutation operation is used for the next 25% of the process trees.
9:         crossoverSet={crossover(tj) | 2αN≤j<3αN }// Crossover operation is used for the next 25% of the process trees.
10:        replacementSet={replacement(tj) | 3αN≤j≤4αN } // The remaining 25% of the process trees are replaced directly.
11:   end foreach
12:        repeatProcessTrees= eliteSelectionSet ∪mutationSet ∪ crossoverSet∪ replacementSet;
13: end foreach
14: PT= selectMAX(repeatProcessTrees)
15: output (PT, oquality(PT)); // Output the best quality process tree.
```

the exclusive choice node is added, as shown in Fig. 6; if an operator node is selected, the entire subtree under the node is reconstructed, as shown in Fig. 7, a sequence operator is selected for a change operation, and the process tree is reconstructed based on the traces (a, b, c) and (b, a, c). First, a and b are selected, parallel operators are then added, and finally, c is added, using sequential operators.

4. The integration of model by genetic algorithm

Algorithm 1 is the pseudo-code of the genetic process mining part in HGSM. Firstly, the overall quality value of each process tree in the population is calculated based on the *OQ* function. Then, according to the proportion of elite selection, select several process trees with the best overall quality and keep them unchanged for the next generation. Genetic operations are used to optimize the quality of the remaining process trees. This process iterates until the stop condition is met, e.g., the number of iterations. Through elite selection and genetic operations, the overall quality of models in each population will gradually improve, and a process tree with the highest overall quality value in the last population is the result of the mining. According to [17], in the process tree population, the proportion of elite selection and three genetic operations was the same, which was 25%.

4. Experiment

4.1 Experimental Preparation

The experiment includes three parts. The first part aims to verify whether the HGSM algorithm can generate the behavior contained in the event log and cluster functionally consistent traces correctly, i.e., components. We use the process log generator (PLG) [22], which can generate events from a specific model. We mine the event log to check if the original model can be generated. In particular, one of the data set in this part is the vending process of the vending machine system (abbreviated as VM). The other is the running process of an online bookstore software system (abbreviated as BS).

Figure 8 is the process model of VM. First, the system displays a product menu, and second, the customer can select the product, and then the system displays the unit price of goods and the customer determines purchase quantity. The system then generates payment information. Next, customers can choose to pay by coins, credit cards or they can cancel their order. Change is also available for coin payment. After payment, the system displays the successful payment by coins or credit card. Finally, the vending machine delivers products. The obtained event log from PLG contains 1000 cases, 8101 events.

The BS data are collected from an online bookstore software system, the system provides a platform for the customers to order books online. After a customer orders some books, these books will be packed and sent for delivery. We created the software operation data by collecting 1000 independent runs that cover all possible operation scenarios. The obtained XES event log contains 1000 cases, with 25960 events.

The online bookstore software system involves three independent components, i.e., the Starter, SearchOffer, and OrderAndDelivery. The Starter component contains the BookstoreStarter class; the SearchOffer component contains the Catalog, BookSeller, and Bookstore classes; the OrderAndDelivery component contains Order classes and Delivery classes. The software starts with the instantiation of the Starter component, then the SearchOffer component obtains an offer for each book, and finally, the OrderAndDelivery component can generate book orders and deliver all the selected books. The UML diagram of the software is shown in Fig. 9. The UML diagram depicts the interactions of classes and methods in one possible software operation, which be used as the ground truth knowledge, to validate the discovery results using HGSM.

The second part of the experiment aims to verify the generated model quality of HGSM. Five quality indicators are used for model quality assessment: replay fitness, precision, generalization, simplicity, and the proposed overall quality. We compare the model quality obtained by the HGSM, with those obtained by state-of-the-art mining algorithms, including the $\alpha\#$ algorithm [24], the IM algorithm,

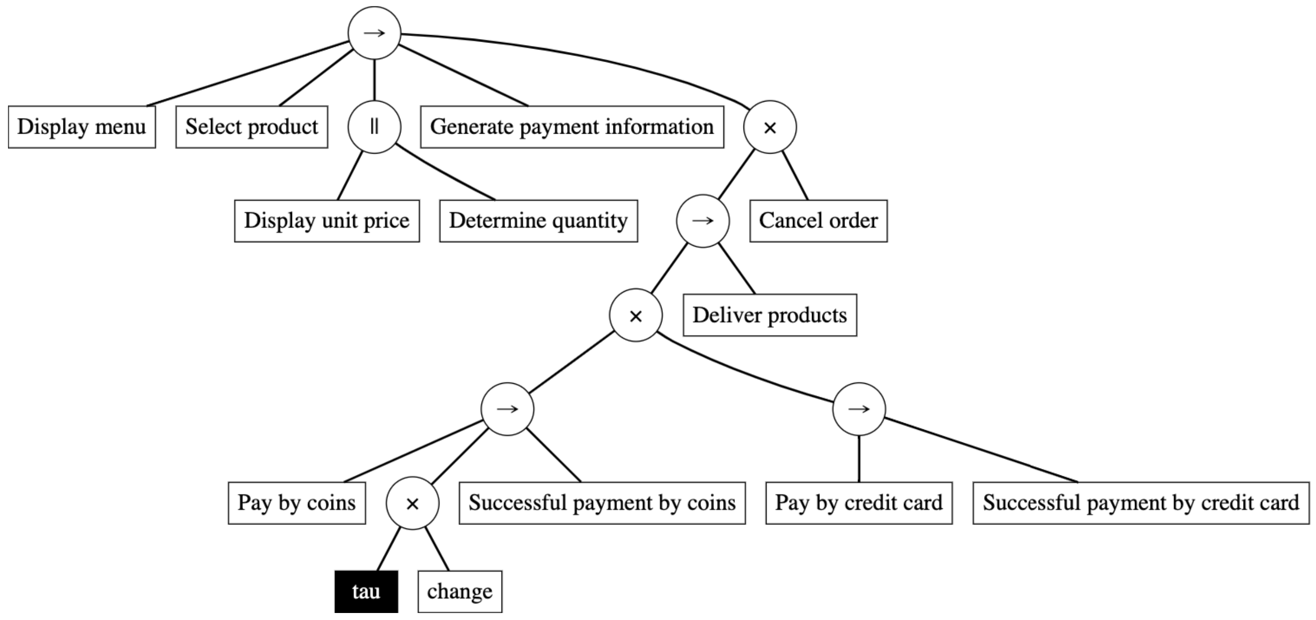


Fig. 8 Vending process of the vending machine

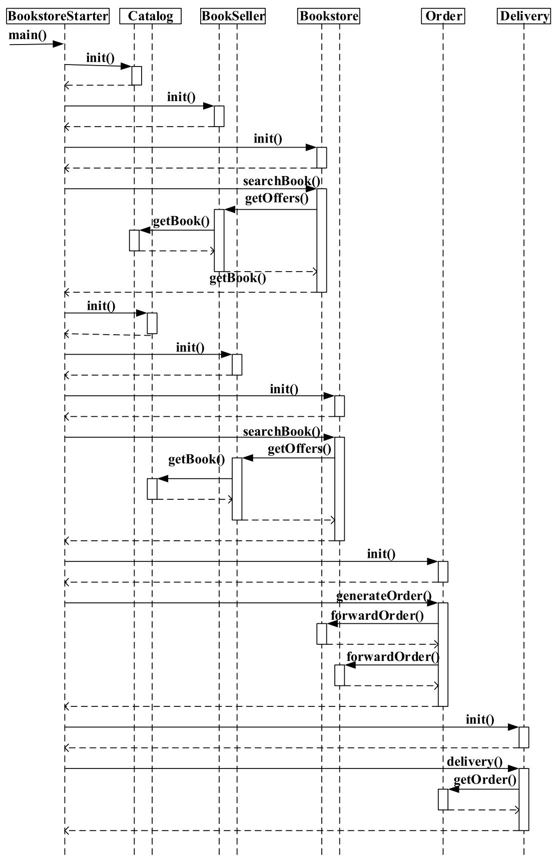


Fig. 9 Sequence diagram of the bookstore software system

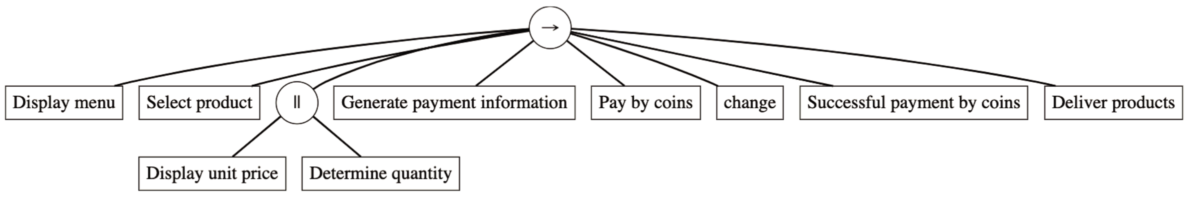
heuristic algorithm [25], integer linear programming algorithm (ILP) [26], the genetic algorithm based on case classification (CGPM) [16], and the GPM algorithm.

Six public data sets are used for the quality validation, some algorithms may discover unsound models for these large-scale data, which leads to the quality of the models cannot be quantified. Thus, the event logs are filtered to ensure soundness. Traffic fine data [27] are the road traffic fine management processes. CCC19 data [28] are the consistency check challenge data. Loan application data [29] are the loan application process data. Representational bias data [30] are the representative bias test data. Benchmarking logs-choose data [31] is the test algorithm scalability data contain the selection loop structure. Benchmarking logs-parallel data [31] is also the scalability data of the test algorithm, which contains a parallel structure. The input problem size was measured by four metrics: number of cases, number of events, number of activities (size of the alphabet), and average case length [7]. The size of the event logs is shown in Table 1.

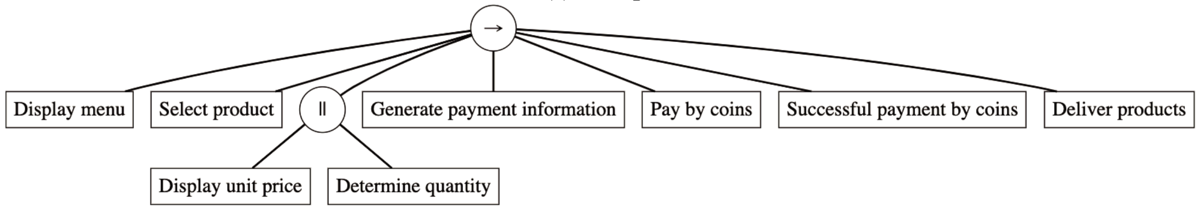
Table 1 Event logs used in the evaluation, with input size statistics

Name	#Case	#Events	#Acts	Avg.|Case|
VM	1000	8108	12	8.1
BS	1000	25960	13	25.9
Traffic fine	489	1710	11	3.5
CCC19	20	1394	58	69.7
Loan application	16	96	6	6
Representational bias	1000	5161	6	5.16
Benchmarking choose	10000	70032	800	7
Benchmarking parallel	10000	107653	80	10

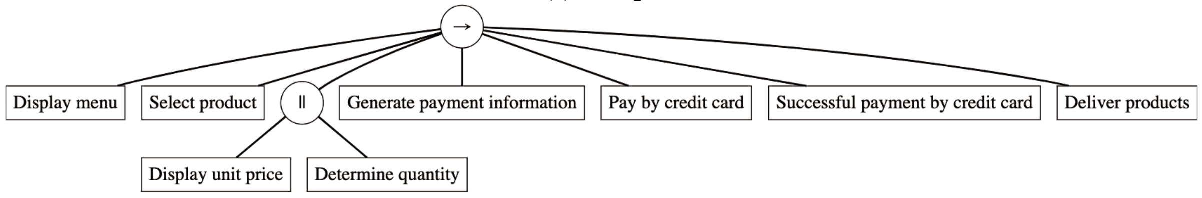
The third part of the experiment compares the convergence speed and the mining consumption time of HGSM and GPM based on the six public data sets. The ratio of four genetic operations and the number of initial populations for two algorithms are the same, and iteration times are set to 1000.



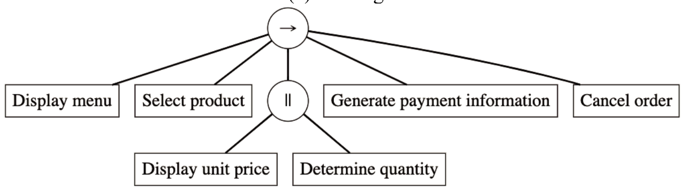
(a) Sub-log1.



(b) Sub-log2.

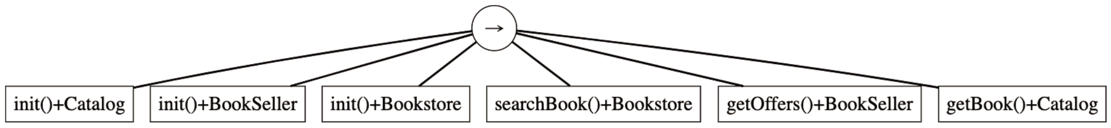


(c) Sub-log3.

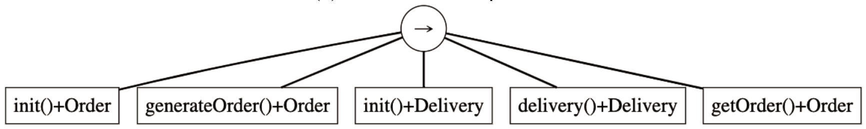


(d) Sub-log4.

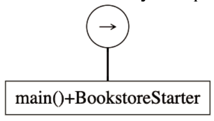
Fig. 10 Sub-logs corresponding models.



(a) Search offer component.



(b) OrderAndDelivery component.



(c) Starter component.

Fig. 11 Sub-models obtained from BS data.

Java SE 1.7.0 67 (64 bit) is used and the experiments are performed on a system with Intel dual-core i7-3770 CPU (3.0GHz) CPU with 12G RAM on Windows 10.

4.2 Experimental Results

4.2.1 Correctness Verification of Mining Results

1. VM data

There are differences in the vending process corresponding to different payment methods, thus we select the trace attribute 'pay type' that represents the payment method as the clustering dimension. Then, we use the IM algorithm to mine the resulting clusters respectively. Figure 10 is the mining results of four sub-logs. It can be seen that the model of sub-log1 is the process of customer pay by coins and system change; the model of sub-log2 is the customer pay by coins and the system does not change; the model of sub-log3 is the process of the customer pay by card; the model of sub-log4 is the customer canceling payment. Therefore,

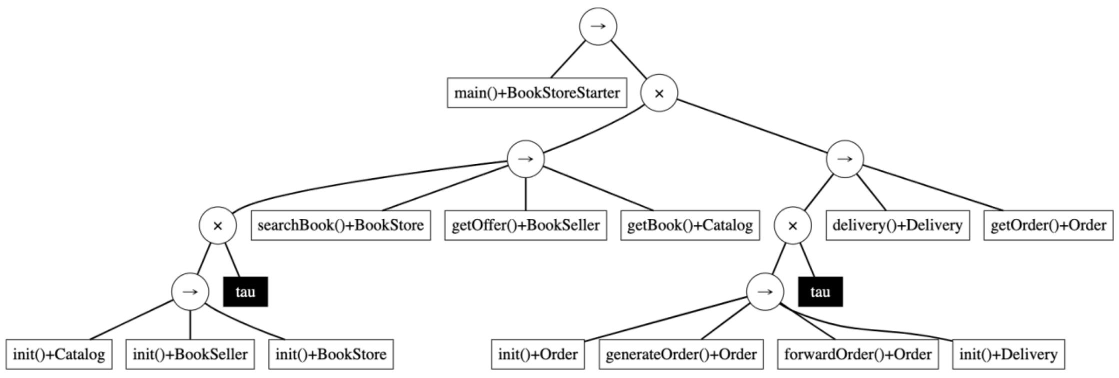


Fig. 12 Final mining result of BS data obtained by HGSM.

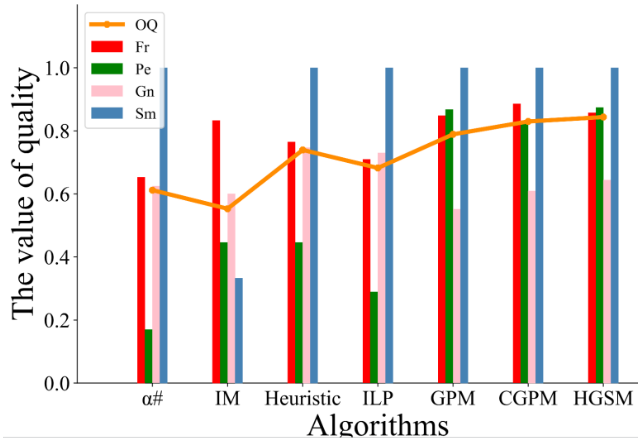


Fig. 13 Road traffic fines.

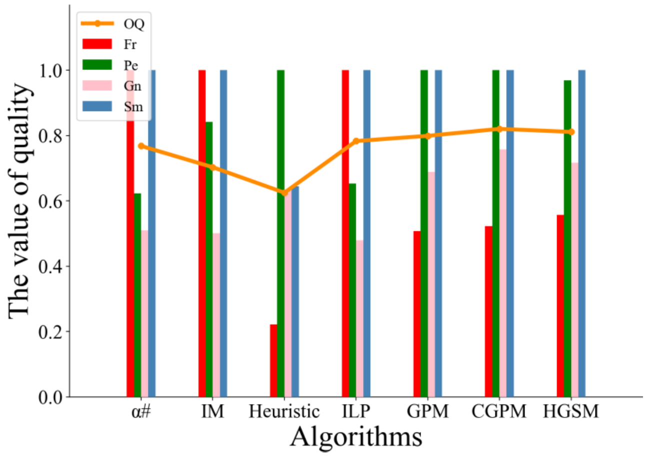


Fig. 14 CCC19.

by selecting the appropriate clustering dimension, we can effectively divide similar traces into the same cluster.

2. BS data

Figure 11 depicts the results of mining the clustered sub-logs of BS data using the IM algorithm, where the events' names are combined in the form of 'method name + class name'. Figure 11 shows that HGSM can distinguish traces from different components. Sub-logs of three components are divided correctly, and three different sub-models represent the operation process of the three components. Figure 12 shows the final mining result, obtained by HGSM, to optimize and integrate all the process trees. By comparison with Fig. 9, we can see that HGSM can mine a model, representing the whole software operation behavior.

4.2.2 Model Quality Analysis

The model quality obtained by different algorithms of six public data sets is shown in Figs. 13–18, the abscissa represents different algorithms, the ordinate represents the quality value of the generated model, and the columns with different colors represent different quality dimensions. These figures indicate that, compared with other mining algorithms, the overall quality of the models generated by HGSM, CGPM, and GPM is higher. This is because the genetic mining algorithms are quality-guided. And compared with GPM, CGPM exhibits higher overall quality owing to the preparation of a higher-quality initial population. And HGSM exhibits the highest overall quality owing to the preparation of a higher-quality initial and the improvement of the mutation operation for GPM.

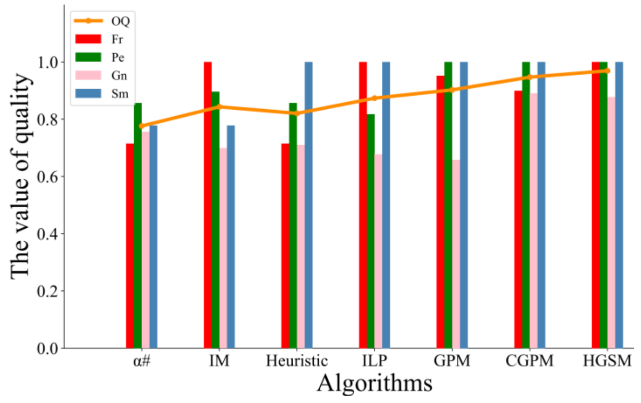


Fig. 15 Loan application.

The model generated by the IM algorithm shows a high value of replay fitness, but its simplicity is low. The heuristic algorithm model has a high value of simplicity, but the value of replay fitness is not ideal. The model obtained by the ILP algorithm can achieve a high degree of replay fitness and simplicity, but the precision and generalization

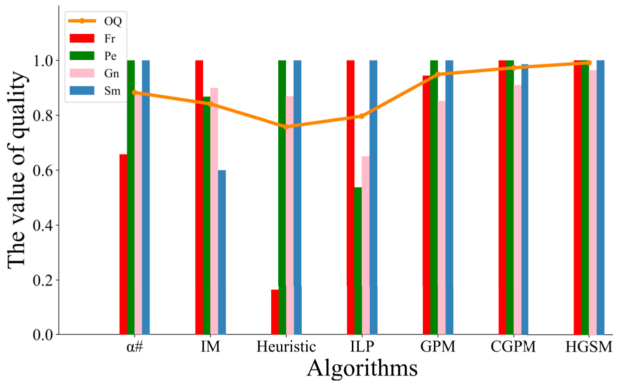


Fig. 16 Representational bias.

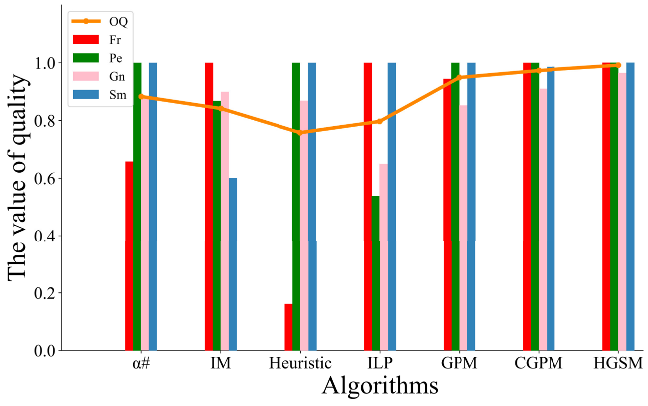


Fig. 17 Benchmarking choose.

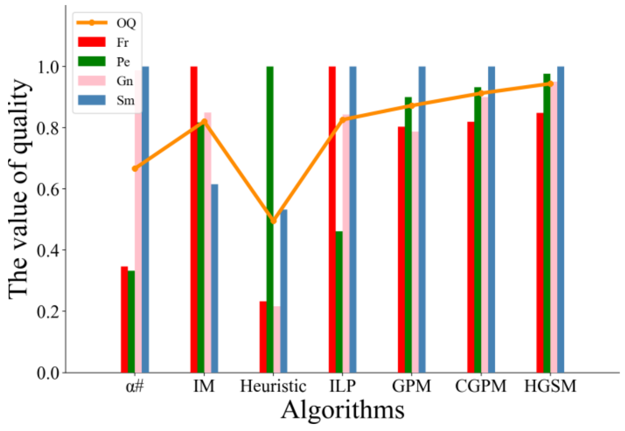


Fig. 18 Benchmarking parallel.

are low. Although each algorithm has its advantages, the values of different quality dimensions of the model mined by the algorithm fluctuate highly, for different data. And outliers occurred in some cases, e.g., the precision value of alpha# and ILP in road traffic fine data, the precision value of heuristic algorithm in CCC19. HGSM uses the OQ function to set the same weight for the four quality indicators, as the mining guide, where it is seen that the four quality dimensions are stable only fluctuate slightly, which can also

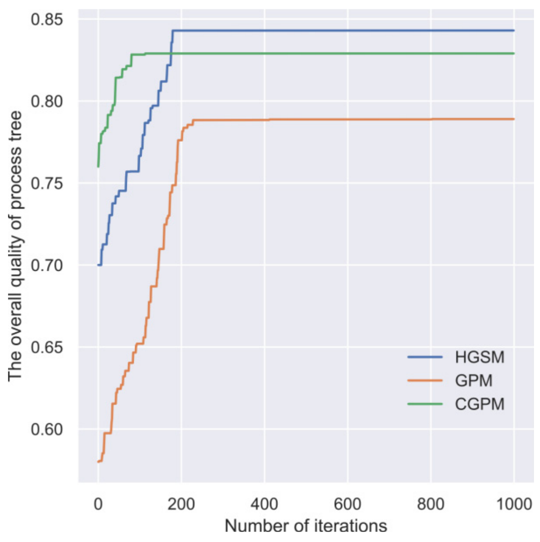


Fig. 19 Road traffic fines

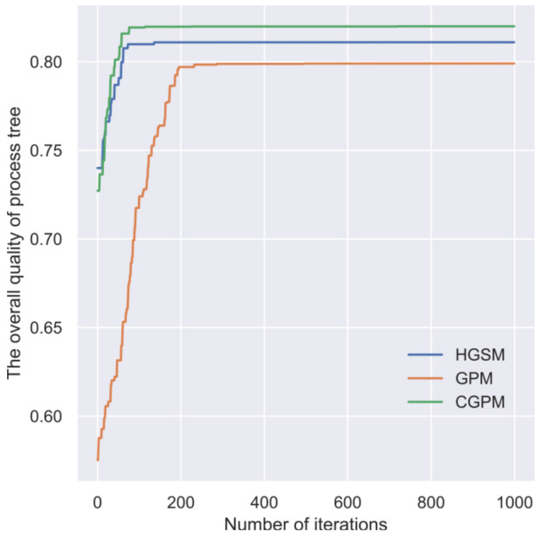


Fig. 20 CCC19.

avoid outliers.

4.2.3 Convergence Speed and the Mining Consumption Time Analysis

During the operation of the genetic algorithm, the overall quality of the model varied with the iteration times of the genetic algorithm. Figures 19–24 show how the overall quality value of the model changes with the number of iterations of six public data sets when mining using GPM, CGPM, and HGSM algorithms. The abscissa represents the number of iterations, the ordinate represents the overall quality value of the process tree. These figures indicate that the overall quality of initial populations randomly generated by the GPM algorithm is mostly less than 0.6. By using the trace clustering and IM algorithm to pre-process, HGSM can improve the overall quality of initial populations, the

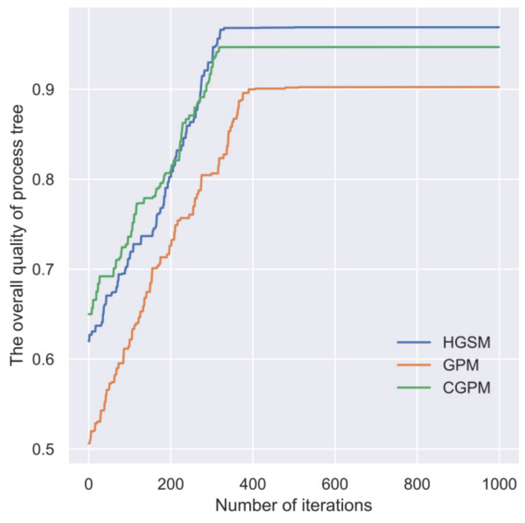


Fig. 21 Loan application.

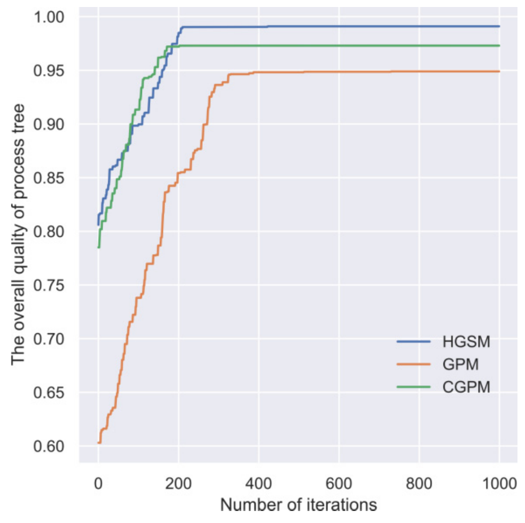


Fig. 22 Representational bias.

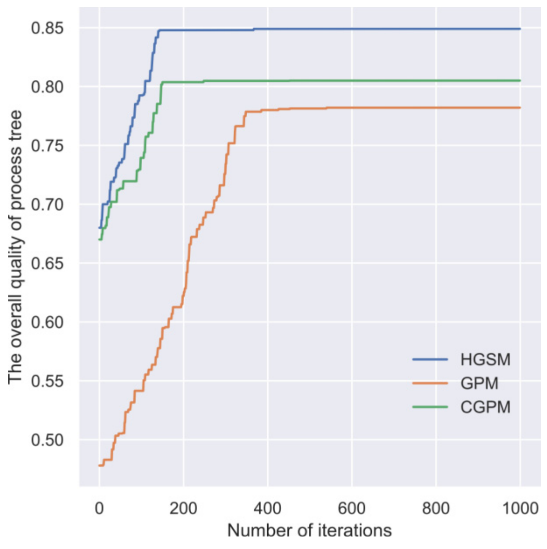


Fig. 23 Benchmarking choose.

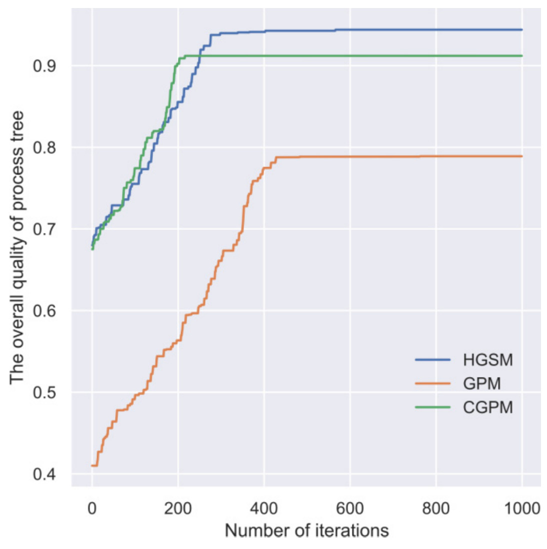


Fig. 24 Benchmarking parallel.

quality of initial populations of HGSM is mostly higher than 0.7. Meanwhile, the convergence of the HGSM algorithm requires fewer iterations than GPM, e.g., For Road traffic fines data, HGSM and GPM converge in 699 and 803 generations, respectively. For Benchmarking choose data, HGSM and GPM converge in 395 and 623 generations, respectively. After convergence, the final model generated by HGSM has higher overall quality than GPM. e.g., For loan application data, the overall quality of the model generated by GPM and HGSM are 0.903 and 0.969, respectively. For Benchmarking choose data, the overall quality of the model generated by GPM and HGSM are 0.783 and 0.849, respectively.

For CGPM, the algorithm uses three different mining algorithms (i.e., alpha, heuristic, and region-based algorithm) to generate three models for each sub log, and put them as the initial population of genetic mining algorithm, this increases the number of individuals of the population, make it easy to get a higher quality process model in the initial population. Therefore, on some data sets, the overall quality of the initial population is higher than that of HGSM, and the convergence speed is faster than HGSM, while after convergence, the final model generated by HGSM has higher overall quality than CGPM, e.g., For Benchmarking choose data, HGSM and CGPM converge in 395 and 600 generations, respectively. For CCC19 data, HGSM and CGPM converge in 630 and 720 generations, respectively. After convergence, the final model generated by HGSM has higher overall quality than GPM. e.g., For Benchmarking choose data, the overall quality of the model generated by HGSM and CGPM are 0.849 and 0.805, respectively. For Benchmarking parallel data, the overall quality of the model generated by HGSM and CGPM are 0.944 and 0.912, respectively. For CCC19 data, although the model quality generated by CGPM is higher than HGSM on this dataset, it is at the expense of enormous time consumption (See Table 2).

HGSM prepares high-quality initial populations for

Table 2 Time-consuming of three algorithms on the six public data sets.

	Traffic fine	CCC19	Loan application	Representational bias	Benchmarking choose	Benchmarking parallel
GPM	4834s	40260s	742s	1948s	242088s	6883s
CGPM	10633s	96523s	1022s	3002s	686951s	7810s
HGSM	**4726s**	**30250s**	**736s**	**1929s**	**195680s**	**5110s**

GPM, the higher the quality of initial populations, the fewer iterations of the algorithm to generate a high-quality model. And the improvement of mutation operation can improve the quality of the models faster and finally accelerate the convergence speed of the algorithm.

Table 2 is the mining time spent of GPM, CGPM, and HGSM of six public data sets. The processing time is only the evolutionary computation part. Table 2 indicates that for smaller data sets, e.g., Loan application and Representational bias, the time advantage of HGSM is not evident compared to GPM and CGPM; while for large data sets, e.g., Benchmarking choose, HGSM can save a lot of time compared with GPM and CGPM. Simultaneously, the time used for mining depends on not only the size of the data but also the structural complexity. The data volume of CCC19 is not large, however, it still takes a long time. This is due to its high structural complexity. For CGPM, for each sub log obtained by classification, the algorithm uses three different mining algorithms to generate models as the initial population of genetic mining algorithm, which increases the number of individuals of the population and greatly increases the processing time of the genetic mining algorithm. Compared with GPM, although CGPM algorithm requires fewer iterations to converge, each iteration requires a long processing time. And this situation is more obvious on large data sets, e.g., Benchmarking choose. In addition, the three algorithms (alpha, heuristic, and region-based algorithm) can not guarantee the soundness of generated model, which needs additional time to process the unsound model. And the three algorithms can not directly generate the process tree. The model can be processed by genetic mining algorithm only after it is transformed into process tree, which will consume additional preprocessing time.

The experimental results demonstrate that HGSM guarantees the correctness of model mining. And by using the trace clustering method, HGSM can divide the complex service-oriented system into multiple functionally independent components correctly. In the respect of model quality, compared with state-of-the-art mining algorithms, HGSM has apparent advantages in terms of the model quality; further, HGSM represents a significant advantage in mining time and the convergence speed compared with the GPM and CGPM.

5. Conclusion

This paper proposes a hybrid genetic service mining method based on a trace clustering population. By optimizing GPM with high-quality initial population preparation, the proposed method can reduce the number of iterations to reach the high-quality model. Meanwhile, by using the information obtained from alignment, we change the mutation operation from random to directed to speed up the population optimization of GPM. The experimental results demonstrate that compared with state-of-art mining algorithms, HGSM has apparent advantages in terms of the model quality; further, HGSM represents a significant advantage in mining time and the convergence speed compared with the GPM and CGPM. Moreover, HGSM can group traces of similar behavior into one cluster, for software operation event logs, HGSM can effectively divide the components, which provides a new solution to the problem of mining large event logs in the current service mining field.

Acknowledgments

This work was supported by Major Project of Science and Technology of Yunnan Province, China (No. 202002AD080002), the Yunnan Provincial Natural Science Foundation, China (No. 202101AT070004, 2019FB135), the Yunnan Province Software Engineering Key Laboratory Open Fund, China (No.2020SE404), the Natural Science Foundation of Yunnan Provincial Department of Education (No. 2019J0942), the Yunnan University "Dong Lu Young-backbone Teacher" Training Program, China (No. C176220200), and the Yunnan Philosophy and Social Science Youth Foundation, China (No.QN2020024).

References

[1] W.M. van der Aalst, "Service mining: Using process mining to discover, check, and improve service behavior," IEEE Trans. Serv,vol.6, no.4, pp.525–535, 2013.

[2] D. Kossmann, F. Leymann, and D. Taubner, "Web Services," Informatik-Spektrum, vol.27, no.2, pp.113–114, 2004.

[3] J. Buijs, B.V. Dongen, and W.V.D. Aalst, "A genetic algorithm for discovering process trees," IEEE C Evol. Comput., pp.1–8, 2012.

[4] S.J.J. Leemans, D. Fahland, and W.M.P.V.D. Aalst, "Discovering block-structured process models from incomplete event logs," Lecture Notes in Computer Science, vol.8489, pp.311–329, 2014.

[5] E. Curry, J. O'Donnell, E. Corry, S. Hasan, M. Keane, and S. O'Riain, "Linking building data in the cloud: Integrating cross-domain building data using linked data," Adv Eng Informatics, vol.27, no.2, pp.206–219, 2013.

[6] D. Sebastian, S. Matthias, M. Martin, and B. Stephan, "Conformance checking: A state-of-the-art literature review," CoRR, 2020.

[7] W.M. van der Aalst, Process Mining: Data Science in Action, Springer, Berlin, 2016.

[8] D. Taibi and K. Systä, "From monolithic systems to microservices: a decomposition framework based on process mining," CLOSER, pp.153–164, 2019.

[9] R. Pérez-Castillo, M. Fernández-Ropero, and M. Piattini, "Business process model refactoring applying IBUPROFEN. An industrial evaluation," J Syst Softw, vol.147, pp.86–103, 2019.

[10] W.M. van der Aalst and H. Verbeek, "Process mining in web services: The WebSphere case," IEEE Data Eng. Bull., vol.31, no.3,

pp.45–48, 2008.

[11] R.P.J.C. Bose and W.M. van der Aalst, "Context aware trace clustering: Towards improving process mining results," SDM, pp.109–20, 2009.

[12] M. Song, C.W. Günther, and W.M.P.V.D. Aalst, "Trace Clustering in Process Mining," BPM, vol.17, no.4, pp.109–120, 2008.

[13] R.P.J.C. Bose and W.M.P.V.D. Aalst, "Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models," BPM, pp.170–181, 2010.

[14] B. Hompes, J. Buijs, W.M. van der Aalst, P. Dixit, and J. Buurman, "Discovering deviating cases and process variants using trace clustering," Proc. 27th Benelux Conference on Artificial Intelligence (BNAIC), pp.5–6, Nov. 2015.

[15] M. Boltenhagen, T. Chatain, and J. Carmona, "Generalized alignment-based trace clustering of process behavior," Petri Nets, pp.237–257, 2019.

[16] L. Yang, G. Kang, W. Cai, and Q. Zhou, "An effective process mining approach against diverse logs based on case classification," 2015 IEEE International Congress on Big Data, IEEE, pp.351–358, 2015.

[17] J.C. Buijs, B.F. van Dongen, and W.M. van der Aalst, "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity," Int J Coop Inf Syst, vol.23, no.01, p.1440001, 2014.

[18] B.F. van Dongen, J. Carmona, and T. Chatain, "A unified approach for measuring precision and generalization based on anti-alignments," BPM, 2016.

[19] D. Fahland and W.M. van der Aalst, "Simplifying discovered process models in a controlled manner," Inf Syst, vol.38, no.4, pp.585–605, 2013.

[20] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. Dongen, and W.M. van der Aalst, "Measuring precision of modeled behavior," Inf Syst E Bus Manag, vol.1, no.13, pp.37–67, 2015.

[21] S.Z. Van, V.B.F. Dongen, W.M. van der Aalst, and H. Verbeek, "Discovering workflow nets using integer linear programming," Computing, vol.100, no.5, pp.375–383, 2018.

[22] A. Burattin, "PLG2: Multi-perspective process randomization with online and offline simulations," BPM, pp.1–6, 2016.

[23] W.M. van der Aalst, A. Adriansyah, A.K.A. de Medeiros, F. Arcieri, T. Baier, T. Blickle T, J.C. Bose, P. van den Brand, R. Brandtjen, and J. Buijs, "Process mining manifesto," Inf. Syst., vol.37, no.3, pp.288–290, 2012.

[24] L. Wen, W.M. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," Data Min Knowl Discov, vol.15, no.2, pp.145–180, 2007.

[25] S. De Cnudde, J. Claes, and G. Poels, "Improving the quality of the Heuristics Miner in ProM 6.2," Expert Syst Appl, vol.41, no.17, 2014.

[26] S.J. van Zelst, B.F. van Dongen, W.M.P. van der Aalst, and H.M.W. Verbeek, "Discovering workflow nets using integer linear programming," Computing, pp.529–556, 2018.

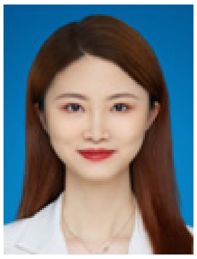
[27] M. de Leoni and F. Mannhardt, "Road traffic fine management process," 4TU ResearchData Dataset, https://www.narcis.nl/dataset/Language/EN/uquery/management/id/12/RecordID/uuid:270fd440-1057-4fb9-89a9-b699b47990f5, 2021.

[28] G. Munoz, JdlF, R. (Rene), "Sepúlveda, M. (Marcos); Fuentes, R. (Ricardo)," Conformance Checking Challenge 2019 (CCC19). 4TU ResearchData Dataset, 2021. https://doi.org/10.4121/uuid:c923af09-ce93-44c3-ace0-c5508cf103ad.

[29] J. Buijs, "Loan application example, configuration 4," 4TU ResearchData Dataset, 2021. https://doi.org/10.4121/uuid:7489b402-bb34-4685-aa02-b002a4ddb698.

[30] W.M. van der Aalst, "Testing Representational Biases," 4TU ResearchData Dataset, 2021. https://doi.org/10.4121/uuid:25d6eef5-c427-42b5-ab38-5e512cca08a9.

[31] W.M. van der Aalst, "Benchmarking logs-parallel," 4TU ResearchData Dataset, 2021. https://doi.org/10.4121/uuid:1cc41f8a-3557-499a-8b34-880c1251bd6e.



Yahui Tang received a B.S degree in computer science and technology from Kunming University of Science and Technology, Kunming, China, in 2016. She is currently a Ph.D. candidate in computer science from Yunnan University. Her main research interests include process mining, business process, and software process.

Tong Li received a Ph.D. degree from De Montfort University, Leicester, UK, in 2007. He is currently a Full Professor with the School of Big Data, Yunnan Agricultural University, China. His research interests include software engineering, concurrent processing, and programming languages.

Rui Zhu received his Ph.D. degree from Yunnan University, Kunming, China, in 2016. He is currently a Lecturer at the school of software, Yunnan University, China. His main research interests include process mining, data mining, and software engineering.

Cong Liu received the Ph.D. degree in computer science and information systems from the Eindhoven University of Technology, Eindhoven, the Netherlands, in 2019. He is currently a full professor in the School of Computer Science and Technology in Shandong University of Technology. His current research interests include business process management, process mining, Petri nets, and emergency management. He is a member of IEEE and IEICE.

Shuaipeng Zhang received his B.S. degree in compute science and technology from Zheng Zhou University, Zheng Zhou, China in 2020. He is currently working towards his M.S degree in Shandong University of Technology. His research interests are in the areas of business process mining.