An Improved Pseudo-parallel Genetic Algorithm for Process Mining

A. Hong Li\textsuperscript{1}, B. Gang Xue\textsuperscript{2}, C. Kunman Li\textsuperscript{3}, and D. Shaowen Yao\textsuperscript{2}

\textsuperscript{1}School of Information Science and Engineering, Yunnan University, Kunming, Yunnan, China
\textsuperscript{2}National Pilot School of Software, Yunnan University, Kunming, Yunnan, China
\textsuperscript{3}School of Computing and Engineering, University of Huddersfield, Queensgate, Huddersfield

Abstract - Process mining is helpful for deploying new business processes as well as auditing, analyzing and improving the already enacted ones. An improved pseudo-parallel genetic algorithm is proposed with an asexual reproduction for avoiding crossover operators’ breach to nice gene patterns. The initial population is produced by greedy algorithm in order to enhance convergence velocity. Information exchange between subgroups employs island model in pseudo-parallel genetic algorithm. These measures are of great significance on reducing complexities and enhancing convergence velocity, as well as increasing global searching ability of the algorithm.

Keywords: process mining, genetic algorithm, parallel, log

1 Introduction

With the rapid development of information nowadays, workflow technology has been widely used. Among them, enterprise modeling is an important field in workflow technology application. Process mining is an important method of Enterprise modeling. It can be used for analyzing, auditing, optimizing the existing system, and help to improve the existing business processes. It can also help enterprises to realize the business process modeling and recycling then enhance the enterprise’s competitive ability.

Research on process mining are mainly focused on mining the heuristic rules based on the binary sequence relationship which was recorded in the logs, and then extract the process model based on these rules. But the structures exist in actual workflow models have many complex structures which increase the difficulty of process mining, such as duplicate tasks, hidden tasks, implicit places, non-free-choices.

The scholars at home and abroad have done a lot of work in the process mining [1, 2]. So far, experts have put forward many meaningful and understandable languages to definite the model. For example, ADONIS model language, Petri nets model language, and the block structure model language. They all have their own characteristics. They have been able to finish the process modeling well basically. In the process mining, Agrawal firstly use the process mining algorithm applied to workflow management system. Initially, this algorithm requirements process does not exist circulation, the model dig out by the process illustrate by diagram which has direction but no circulation. But Agrawal disposed part of the circulation problem when he improved the algorithm. Herbst and others proposed another algorithm [7, 8] based on the ADONIS language. This algorithm deal well with the structure which containing the same activity name in the process, and find simple circulation structure. But to finding hide structure it haven’t realized. Aalst and Weijter posed \(\alpha\) algorithm [10, 11] that based on the extended Petri nets to find equivalent substitution with And-split , And-join structure. But it only uses for the relationship between each activity relationship. Thus, it cannot find the structures which need considering the global structure like NFC (no-free-choice) structure. And also it is helpless in two conditions that the same process contains the same activity name or complex circulation structures.

The purpose of the process mining from existing event log is to automatically find valuable information. The first dedicated process mining algorithms were proposed in [1][2]. The discovered information can be used to deploy new systems that support the execution of business processes or as an analytical tool that helps in auditing. At present many domestic and foreign scholars devoted to the research of process mining. A lot of people dig workflow model using WF nets, and based on this, puts forward \(\alpha\) algorithm, \(\beta\) algorithm. The limitations [3] of \(\alpha\) algorithm cannot mining the complex structure, such as loop, invisible task, non-free-choice constructs and so on. \(\alpha^+\) algorithm, \(\alpha++\) algorithm extended some function based on it and solved part problem, but they did not consider the interval between the start events and end events. The \(\beta\) algorithm considers the interval, but cannot mine the looped and invisible tasks.

2 Related works

Since the mid-nineties several groups have been working on techniques for process mining [4], [5], [10] discovering process models based on observed events. In [3] an overview is given of the early work in this domain. The idea to apply process mining in the context of workflow processes was introduced in [5]. In parallel data [12] looked at the discovery of business process models. Cook et al. investigated similar issues in the context of software engineering processes [10]. Herbs was one of the first to...
tackle more complicated processes, which contain duplicate tasks.

Most of the classical approaches have problems in dealing with concurrency. The algorithm [4] is an example of a simple technique that takes concurrency as a starting point. However, this simple algorithm has problems in dealing with complicated routing constructs and noise (like most of the other approaches described in literature). In [14] a more robust but less precise approach is presented. Heuristics or genetic algorithms [13] have been proposed to deal with issues such as noise.

Genetic algorithm (GA) is a kind of random search method reference the law of the biological evolution, its main algorithm steps are as Fig.1.

![Fig.1. Genetic algorithm](image)

Main steps of our genetic algorithm:

- Step one: Read the event log.
- Step two: Calculate the dependency relations among activities.
- Step three: Build the initial population.
- Step four: Calculate individuals’ fitness.
- Step five: Stop and return the fittest individuals
- Step six: Create next population by using the genetic operators.

Genetic algorithm has been widely used in many engineering fields such as complex function optimization, structure design, system control, machine learning, and image processing and so on. It has global searching ability, high efficiency of group search strategy, irrelevance of the goal of gradient information and simple universal. But it has existent problem like the slow of efficiency and resource consumption. There put forward a Pseudo-Parallel Genetic Algorithm with asexual reproduction, it reduces the complexity of the algorithm, improves the algorithm convergence speed and improves the operation efficiency of algorithm.

3 Foundation technologies

3.1 Population initialization

First of all we should define the causal matrix.

Definition 1. A causal matrix is a tuple \( CM = (A,C,I,O) \), where: \( A \) is a finite set of activities, \( C \subseteq A \times A \) is the causality relation, \( I \in A \rightarrow p(p(A)) \) is the input condition function, and \( O \in A \rightarrow p(p(a)) \) is the output condition function; such that: \( C = \{(a1,a2) \in A \times A | a1 \in U I(a2) \} \), \( C \cup \{(ao,ai) \in A \times A | ao = \emptyset \land ai = \emptyset \} \) is a strongly connected graph.

The initial population is randomly built by the genetic algorithm. When building the initial population, we roughly follow Definition 1. Given a log, all individuals in any population of the genetic algorithm have the same set of activities (or tasks) \( A \). This set contains the tasks that appear in the log. However, the causality relation \( C \) and the condition functions \( I \) and \( O \) are randomly built for every individual in the population. As a result, the initial population can have any individual in the search space defined by a set of activities \( A \). Note that the higher the amount of tasks that a log contains, the bigger this search space.

3.2 Fitness Calculation

If an individual in the genetic population correctly describes the registered behavior in the event log, the fitness of that individual will be high. In our approach the fitness is strongly related to the number of correctly parsed traces from the event log. Note that in case of noisy situation, we cannot aim at mining a process model that can correctly parse all traces, because the traces with noise cannot also be parsed by the desired model.

We use the naive semantics with silent transitions that only fire when needed and simply play the “token game”. When the activity to be parsed is not enabled, the parsing process does not stop. The problem is registering and the parsing proceeds as if the activity was enabled (conceptually, this is equivalent to adding the necessary missing tokens in the Petri net to enable the activity and, then, firing it). We adopt this parsing semantics because it is more robust to noisy logs and it gives more information about the fitness of the complete process models. In a noise-free situation, the fitness of a model can be 1 (or 100%) (i.e., all traces can be parsed). In practical situations, the fitness value ranges from 0 to 1. The exact fitness of an individual to a given log is given by the formula:

\[
\text{Fitness} = 0.60 \times \frac{\text{allProperlyCompletedLogTraces}}{\text{numberOfTracesAtLog}} + 0.40 \times \frac{\text{allParsedActivities}}{\text{numberOfActivitiesAtLog}}
\]

\[ (1) \]

\text{numberOfActivitiesAtLog} is the number of activities in the log, \text{numberOfTracesAtLog} is the number of log traces, \text{allParsedActivities} is the sum of parsed activities for all log traces, \text{allProperlyCompletedLogTraces} is the number of log traces that were properly parsed.

3.3 Stochastic tournament selection operator

To select a parent, the tournament selection algorithm randomly selects 5 individuals and returns the fittest individual among the fiveones.
3.4 Asexual reproduction

Here, the sexual reproduction crossover operator as a genetic operator may not meet certain constraints, so we must use "fix" algorithm or some special crossover operator such as partial match cross (PMC), order crossover (OC) and circular cross (CC). But these crossover operators often have large damage to good genes mode. It can't be avoided to increase the complexity of the algorithm. So, in this paper use the asexual reproduction; It is said that to cancel the crossover operator, replaced by gene recombinant operator on individual. It simplifies the process of genetic operations, raises the calculation efficiency, and does not require the initial population diversity; there isn't a premature convergence problem.

3.5 Gene reconstruction

Adjusted gene symbol that Swap mutation, Insert mutation and Invert mutation the Relative position of chromosome be called gene recombinant operator. Individual fitness function calculation method can be further simplified because using asexual reproduction, that reorganization value is equal to the individual's fitness before the reorganization of fitness value plus the value of the fitness changed.

3.5.1 Swap mutation

Swap mutation is the exchange of encoded string of individual loci between two randomly selected gene values, resulting in a new tour route, for example:

\[ T_X = (1,2,\ldots,i - 1,i,i + 1,\ldots,j - 1,j,j + 1,\ldots,n - 1,n) \rightarrow (1,2,\ldots,i - 1,j,i + 1,\ldots,j - 1,i,j + 1,\ldots,n - 1,n) = T'_X \]

3.5.2 Insert mutation

Insert mutation is at first elected randomly two loci in the individual encoding string, then put a gene of loci after another gene loci. For example:

\[ T_X = (1,2,\ldots,i - 1,i,i + 1,\ldots,j - 1,j,j + 1,\ldots,n - 1,n) \rightarrow (1,2,\ldots,i - 1,i,j,i + 1,\ldots,j - 1,j + 1,\ldots,n - 1,n) = T'_X \]

3.5.3 Invert mutation

Invert mutation is reverse order the gene that selected randomly between the two loci of individual coding string, resulting in a new tour route, for example:

\[ T_X = (1,2,\ldots,i - 1,i,i + 1,\ldots,j - 1,j,j + 1,\ldots,n - 1,n) \rightarrow (1,2,\ldots,i - 1,j,j + 1,\ldots,i + 1,i,j + 1,\ldots,n - 1,n) = T'_X \]

3.6 Island model

Pseudo-Parallel Genetic Algorithm is that run multi group parallel genetic algorithm on one processor. This article will initial population into three sub groups. Each child group independently evolves according to certain mode, at the right time, sub group can change some information, which can maintain the diversity of the population, and keep the different evolution direction, so as to suppressing precocious phenomena. Island model is also called coarse-grained model. Transfer strategy is the operator that changes information between sub groups. The migration can speed up in the propagation of the individual groups, improve convergence speed. This article uses the migration model as below in Fig.2.

![Fig.2. migration model](image)

3.7 Stop criteria

The mining algorithm stops when (i) it finds an individual with fitness equals 1; or (ii) it computes \( n \) generations, where \( n \) is the maximum number of generation that is allowed; or (iii) the fittest individual has not changed for \( n/2 \) generations in a row.

4 Genetic Operations

We use elitism, crossover and mutation to build the individuals of the next generation. A percentage of the best individuals (the elite) are directly copied to the next population. The other individuals in the population are generated via crossover and mutation. Two parents produce two offspring. To select parents, a tournament is played in which five individuals in the population are randomly drawmand the fittest one always wins. The crossover rate determines the probability that two parents undergo crossover. Crossover is a genetic operator that aims at recombining existing material in the current population. In our case, these materials are the set of current causality relations in the population. The crossover operation should allow the complete search of the space defined by the existing causality relation in a population. Given a set of causality relations, the search space contains all the individuals that can be created by any combination of a subset of the causality relations in the population. Thus, our crossover operator allows an individual to: lose tasks from the subsets in its I/O condition functions (butnot necessarily causality relations because a same task may be in more than one subset of an I/O condition function), add tasks to the subsets in its I/O condition functions (again, not necessarily causality relations), exchange causality relations with other individuals, incorporate causality relations that are in the population but are not in the individual, lose causality relations, decrease the number of sub-sets in its I/O condition functions, and/or increase the number of subsets in its I/O condition functions. The crossover point of two parents is a randomly chosen task. Note that, after crossover, the number of causality relations for the whole population remains constant, but how these relations appear in the offspring maybe different from the parents.
After the crossover, the mutation operator takes place. The mutation operator aims at inserting new material in the current population. In our case, this means that the mutation operator may change the existing causality relations of a population. Thus, our mutation operator performs one of the following actions to the I/O condition functions of a task in an individual: (i) randomly choose a subset and add a task (in A) to this subset, (ii) randomly choose a subset and remove from this subset, or (iii) randomly redistribute the elements in the subsets of I/O into new subsets. Every task in an offspring may undergo mutation with the probability determined by the mutation rate.

Improved pseudo parallel genetic algorithm process description as follows:

Step1: Initialize the genetic algebra counter: \( t \leftarrow 0 \)
Step2: Create initial group \( P(t) \) based on the method of greed, perform preprocessing to \( P(t) \) and individual fitness calculation.
Step3: Copy \( P(t) \) into three son-group: \( P(t) = P_1(t), P_2(t), P_3(t) \), perform three different genetic restructuring operator, swap mutation, insert mutation, and invert mutation respectively.
Step4: \( P_i(t)(i = 1, 2, 3) \) evolution independent:
\( P'_i(t) \leftarrow \text{Selection}[P_i(t)](i = 1, 2, 3) \); do variation operation by gene recombinant operator
\( P''_i(t) \leftarrow \text{Mutation}[P'_i(t)](i = 1, 2, 3) \).
Step5: Group calculation individual fitness in different \( P''_i(t)(i = 1, 2, 3) \).
Step6: Change the information between different \( P''_i(t)(i = 1, 2, 3) \) by transfer strategy, and get the offsprings
\( P_i(t + 1) \leftarrow \text{Exchange}[P''_1(t), P''_2(t), P''_3(t)] \).
Step7: Judge the conditions for the termination:
If do not meet the conditions for the termination then \( t \leftarrow t + 1 \), go to Step4;
If meet the termination conditions, the optimal results, the output is over.

5 Experiment

As a first test of our Pseudo-Parallel Genetic Algorithm (PPGA), we applied it for noise-free event logs and checked if it could mine process models that contain all the behavior in these logs. In other words, the mined model should have the fitness \( F = 1 \). During the experiment, the genetic algorithm mined event logs from nets that contain 5, 7, 8, 12 and 22 tasks. These nets contain short loops, parallelism and/or non-free-choice constructions. Every event log has 1000 random executions of the nets. For each noise-free event-log, 10 runs of the genetic algorithm were executed. The populations had 500 individuals and were iterated for at most 100000 generations. The crossover rate was 1.0 and the mutation rate was 0.01. The elitism rate was 0.01. The initial population might contain duplicate individuals.

The generic mining algorithm presented in this paper is supported by a plugin in the ProM framework. Fig.3 shows screenshot of the plugin.

Fig.3. result

6 Conclusion

In this paper we presented our first experiences with a more global mining technique. The experiment shows that Pseudo-Parallel Genetic Algorithm can mining the process very well. These measures are of great significance on reducing complexities and enhancing convergence velocity, as well as increasing global searching ability of the algorithm.

7 ACKNOWLEDGEMENT

This paper is supported by Laboratory of Network Intelligent Computing @ Yunnan University.
8 References


