Evolutionary process discovery

Dr. Laurens Bliek – Dr. Hendrik Baier – Bart von Meijenfeldt – Tijmen P. Kuijpers

Processes and Petri Nets

The financial institute PETRINAS manages loan requests from its customers according to a not well-formalized process model. The process starts when a request is received. Then, the request passes preliminary assessments, aimed at verifying whether the applicant meets the requirements. The request also undergoes frauds detection. As soon as an application is not eligible, the process is terminated. The user receives an offer and can communicate whether he/she intends to accept it. If this is not the case, the process ends. Otherwise, a contract is drawn and finally sent to the customer. The actual sequence of these actions is unknown to the institute.

A sequence of actions can be formalized and investigated using a Petri Net (PN). A PN is a directed bipartite graph characterized by two sets of nodes: the **places** (used to represent the status of the process) and the **transitions** (corresponding to the activities described above) connecting the places. Places can only be connected to transitions, and transitions can only be connected to places.

PNs can be effective to represent and study processes. For example, the PN shown in Fig. 1 represents one possible action: receiving the request, a transition which brings from place p1 to place p2.

Fig. 1 - Example of Petri net with two places and one transition.

Given a set of M places and a set of N transitions, we can represent the PN's connections using a matrix A of dimension $N \times 2$ with values {0, 1, ..., M – 1}.

The m-th row in the matrix denotes the m-th transitions. The 1st column represents the origin place and the 2nd column the destination place of the transitions.

- If $a_{m,0}$ = 3 this means that the m-th transition has an incoming connection from place 3
- If $a_{m,1}$ = 5 this means the m-th transition has an outgoing connection to place 51

As an example, consider the PN in Fig. 2.

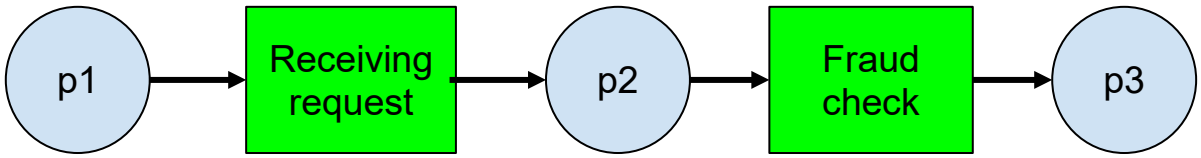

Fig. 2 - Example of Petri net with three places and two transitions.

The PN in Fig. 2 can be represented by the following 2×2 matrix:

$$\begin{matrix} 0 & 1 \\ 1 & 2 \end{matrix}$$

Companies try to log the activities of their processes as well as possible. Using these logs, one can attempt to deduce the corresponding PN. For instance, see the dataset underneath that represents 3 traces of a process.

"Receiving request", "Fraud check"
"Receiving request", "Fraud check"
"Receiving request", "Fraud check"

As all of these traces are the same, one can deduce that the simple PN of Fig. 2 represents the process best.

The company PETRINAS has collected a dataset composed of 100 traces of this kind from their processes (contained in the file dataset_a1.txt that you can find in Canvas) and is now interested in creating and investigating the associated PN. The company knows that there are 9 places and 10 clear activities + 2 "invisible activities" in their process. These invisible activities do not show up in the logs, but do affect the place of the process instance. This leads to an unknown matrix with dimension 12×2 to be discovered. As each element of the matrix can take values {0, 1, ..., 8}, then the number of possible PNs to be tested is $9^{24} \approx 8 \times 10^{22}$. Considering that the evaluation of the alignment between a PN and a dataset is relatively expensive, it is clearly not feasible to solve this through an exhaustive search.

Genetic Algorithms to the rescue

We can explore this huge search space using a **Genetic Algorithm** (GA). The candidate solutions (i.e., the PN matrices) can be encoded as **linearized lists of integer numbers**. For instance, the PN in Fig. 2 can be encoded as: [0, 1, 1, 2].

To quantify the fitness of the PN with the trace log we will be using the concept of alignments. The core idea is to match each of the traces from the logs to traces to the most similar trace that could be created using the PN. Often these similar traces will not perfectly match the logged traces, as sometimes unexpected events happen which can only be captured by a very large model. Therefore, to find the best PN non-perfectly aligned traces will carry a cost which is based on the difference in the activities between the traces. Where each activity present in one trace but not the other adds a penalty. Invisible activities are excluded from these costs,

as they will never be found in the logs. Full details can be found in Wil van der Aalst's *Process Mining Data Science in Action* chapter 8.3. To help you with this assignment, PETRINAS has already implemented a fitness function based on alignments and some hand-crafted domain knowledge for you.

For the assignment, you have to perform multiple tasks. **These tasks will require about 10 hours of computing time. Make sure to plan ahead for this.** Tweak the computational requirements of your runs in such a way as to (approximately) use the computing budget specified in each of the tasks. Before you start with the assignment, install the required packages as described in Appendix A.

The first task is to deal with this problem with GA:

1. (5 points) implement the GA using deap.algorithms.eaSimple method in DEAP. Please use ['cxTwoPoint', 'mutFlipBit', 'selTournament'] as the [Crossover, Mutation, Selection] operators, record the default parameter of your initial GA and plot the best fitness at each iteration ; **Computing time required: 2 minutes**

Then you have to **investigate the performance** of the GA: in task 2 and 3 that you will do, check the **distribution of the fitness values** of the best solution found across 10 runs by creating **boxplots**. Also, check the **average best fitness**(ABF) at each iteration (over 10 runs) to verify possible premature convergence by creating line plots. The Figure 3 is an example of multiple line plots and boxplots for reference (please notice that we have a maximization problem here and the line plot might look different).

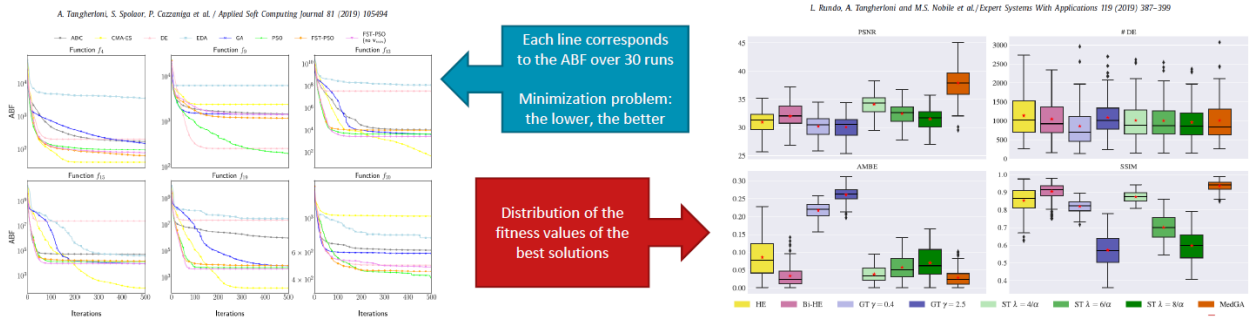

Fig. 3 - Example of line plots and boxplots.

2. (5 points) **compare the performance of possible variants of your GA**, e.g., using different crossover, mutation, and selection operators. Test the variants in the table below, and investigate both the impact to the **optimization performances** and the overall **running time**, keep the best choice according to ABF for the later questions**.** You can refer to the operators described in DEAP documentation: Operators; **Computing time required: 240 minutes**

Crossover	Mutation	Selection	ABF	Time (Seconds)
cxTwoPoint	mutFlipBit	selTournament		
cxTwoPoint	mutFlipBit	selRoulette		
cxTwoPoint	mutShuffleIndexes	selTournament		

cxTwoPoint	mutShuffleIndexes	selRoulette
cxOrdered	mutFlipBit	selTournament
cxOrdered	mutFlipBit	selRoulette
cxOrdered	mutShuffleIndexes	selTournament
cxOrdered	mutShuffleIndexes	selRoulette

3. (5 points) investigate the **impact of mutation and crossover probabilities** and try to determine the **best choice of hyper-parameters** for this problem, keep the best choice for the later questions. You are suggested to test mutation and crossover probabilities in the table below; **Compute time required 320 minutes**

Average Best Fitness		crossover probabilities			
		0.2	0.4	0.6	0.8
mutation probabilities	0.2				
	0.4				
	0.6				
	0.8				

4. (3 points) if you inspect the supplied fitness function you can see that PETRINAS has encoded some domain knowledge into their fitness function. It would be more efficient not to have evaluate such solutions, but instead replace them with more promising candidates. In this task, you are tasked to exploit tool decoration to handle such type of **constraints** and "fix" the candidate solution after a mutation and/or crossover to comply with PETRINAS domain knowledge. As it can be quite complex to solve all constraints at once, you only have to make sure that the new candidates comply with the third and fifth constraint. These constraints prevent self-loops and activities to go back in place. You are allowed to use randomness to generate new candidates, but you must make sure they comply with the constraints.
5. (2 points) Combine the best selected hyperparameters from task 2 and 3 and the tool decoration from task 4 to run GA for approximately 60 minutes. Then use the function matrix_to_graphviz() provided in the file support.py to **plot the best Petri net** both before and after using tool decorator (best with respect to the fitness function) and **briefly comment on it** for PETRINAS. The function receives as argument a candidate solution, in the form of a linearized matrix. The function creates a pdf file containing the plot. In order to use the matrix_to_graphviz() you have to **install the required libraries**: pm4py and graphviz (see the instructions in the appendix of this file); **Computing time required 30 minutes**

Summarize all your findings into a PDF report:

- explain **which tests** you have performed;
- show the **figures** and motivate which was the **best configuration** of the GA according to your tests.

Remember: submit both the **report** and the **source** Python code. If one of the two files is missing, the score will be 0 points.

The max length of the report is: **6 pages**.

If you have any comments, questions, or doubts about the assignment, please **open a thread on Canvas Discussions**: other students may be looking for the same answer. Also, feel free to answer your colleagues. **We will not answer private emails.**

Finally, please remember that this is just an exercise about GAs: it is not important if the PNs do not look completely reasonable as long as they are optimal with respect to the fitness function.

Appendix A: how to install required packages and graphviz

To install the packages (except for **graphviz**) simply use pip:

pip install pm4py pandas deap

To install **graphviz** please follow the instructions on the website:

https://graphviz.org/download/

There are executable installation files for the main operating systems. In order to make graphviz available to pm4py, **please make sure that graphviz's executable files are in your system's path**. The installer should ask you about this: please answer yes. You may need to restart your laptop after the installation of graphviz.

Appendix B: Assignment Rubric

Q1 (5 points)	• The code of the initial GA is functional and well-documented; • The plot of the best fitness at each iteration is provided.
Q2 (5 points)	• The code for testing different operators is functional and well-documented; • The table is filled out properly; • The boxplot used to show the distribution of the best fitness values is provided; • The line plot of ABF at each iteration is provided.
Q3 (5 points)	• The code for hyper-parameters tuning is functional and well-documented; • The table is filled out properly; • The boxplot used to show the distribution of the best fitness values is provided; • The line plot of ABF at each iteration is provided.
Q4 (3 points)	• The code for tool decoration is functional (the solution meets the constraint in Q4 description) and well-documented;
Q5 (2 points)	• The plots for the best Petri nets are provided; • The report describes the findings about the best Petri nets and makes some comments.