Project 1:
Process Mining Using Expectation-Maximization with Gaussian Filtering

Process Mining Using Expectation-Maximization with Gaussian Filtering

Background:

The goal of process mining is to *turn event data into insights and actions* by constructing a process model. At the minimum, it requires case id, activity, and timestamp (Fig. 1).

If the case id is missing, a process model can be estimated using a probabilistic *Expectation-Maximization Procedure* [1]. The quality of this method can be negatively affected by real-life data that contain infrequent or chaotic activities [2]. Infrequent activities can manifest as short processes.

Case	Activity	Timestamp	Resource	Customer
...	...	...	...	...
pizza-56	buy ingredients (bi)	18:10	Stefano	Valentina
pizza-57	buy ingredients (bi)	18:12	Stefano	Giulia
pizza-57	create base (cb)	18:16	Mario	Giulia
pizza-56	create base (cb)	18:19	Mario	Valentina
pizza-57	add tomato (at)	18:21	Mario	Giulia
pizza-57	add cheese (ac)	18:27	Mario	Giulia
pizza-56	add cheese (ac)	18:34	Mario	Valentina
pizza-56	add tomato (at)	18:44	Mario	Valentina
pizza-56	add salami (as)	18:45	Mario	Valentina
pizza-56	bake in oven (bo)	18:48	Stefano	Valentina
pizza-57	add salami (as)	18:50	Mario	Giulia

Figure 1. A sample event log.

Objective:

Remove short processes by extending the standard Expectation-Maximization algorithm to include *Gausian filtering for low transition probabilities*.

株式会社 日本AIコンサルティング
github.com/johnisaacenriquez

[1] Ferreira and Gillblad. Discovering Process Models from Unlabelled Event Logs (2009) https://doi.org/10.1007/978-3-642-03848-8_11
[2] Tax et al. Discovering more precise process models from event logs by filtering out chaotic activities. (2019) J Intell Inf Syst **52**, 107–139.

Process Mining Using Expectation-Maximization with Gaussian Filtering

Methodology:

In Expecctation-Minimization algorithm, the transition matrix M and source sequence s can be extimated iteratively from a given symbol sequence x. For each iteration, we modifiy the transition matrix by filtering the transition probabilities that are *lower than the threshold* value. For row a in the transition matrix M_a threshold t_a is defined as:

$$\bar{M}_a = \sum_{b=1}^{B} \frac{1}{B} M_{a,b}$$

$$\sigma_a = \sqrt{\frac{1}{B} \sum_{b=1}^{B} \left| M_{a,b} - \bar{M}_a \right|}$$

$$t_a = \bar{M}_a - s\sigma_a$$

where $M_{a,b}$ is the transition probability of activity a to b, B is the total number of unique activities, $\bar{M}_a$ and σ_a are the mean and standard deviation of the matrix elements for row a, and s is the standard deviation parameter. Transition probabilities lower than the threshold is filtered out by setting to zero.

Process Mining Using Expectation-Maximization with Gaussian Filtering

Results:

The modified algorithm is tested on a test log provided by J-AIC for July 2022. Process mining is performed for 4 trials with the following standard deviation factors:

Trial 1	No Filtering	
Trail 2	Low Filtering	$s = 3.0$
Trail 3	Mid Filtering	$s = 2.0$
Trail 4	High Filterinng	$s = 1.0$

The process mining results are evaluated by the average length of process (Fig. 2) and percent long processes (Fig.3) defined as:

$$percent\ long\ processes = \frac{processes\ longer\ than\ 4\ activities}{total\ number\ of\ processes} \times 100\%$$

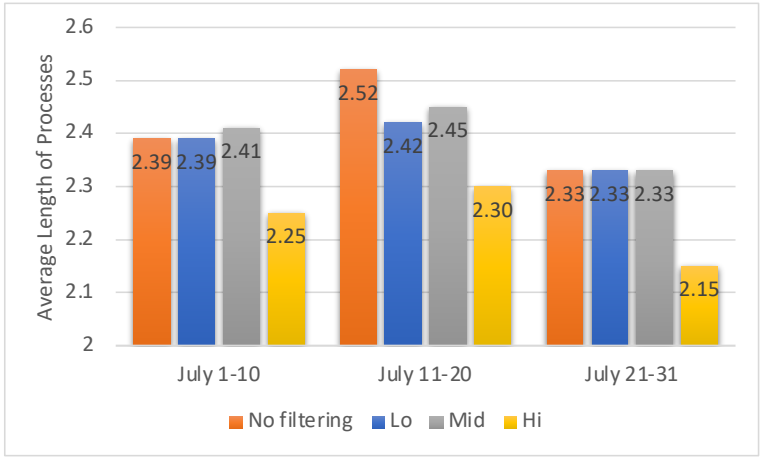

Figure 2. Comparison of average length of process resulting from various levels of filtering.

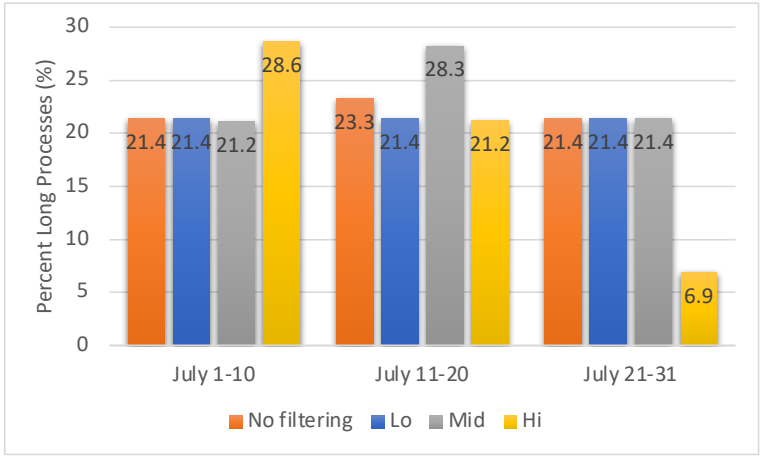

Figure 3. Comparison of percent long processes resulting from various levels of filtering.

Process Mining Using Expectation-Maximization with Gaussian Filtering

Results:

The average length of processes for July 1-10 is increased with mid filtering ($s = 2.0$). The percent long processes for July 1-10 and July 11-20 are increased with hi filtering ($s = 1.0$) and mid filtering ($s = 2.0$), respectively.

Conclusions:

Gaussian filtering can be used to *remove low probability transitions*. With optimal s parameter values, this can lead to an *increase in the average length of processes and percent long processes*. However, the results are *highly sensitive* to the input data.

Recommendation:

Optimization of the s parameter is *necessary* to for best results.

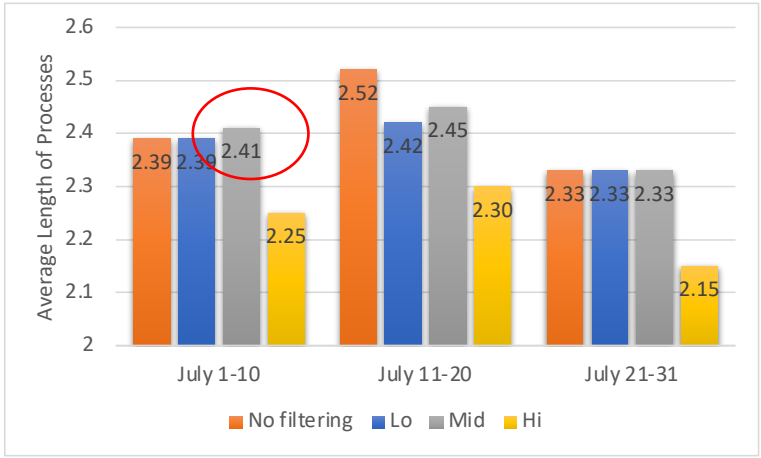

Figure 2. Comparison of average length of process resulting from various levels of filtering.

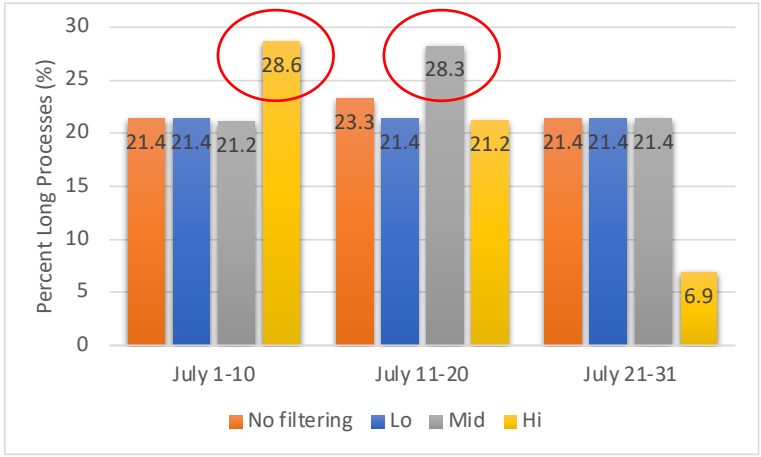

Figure 3. Comparison of percent long processes resulting from various levels of filtering.

Project 2
Process Mining Using Natural Language Processing Neural Network

Process Mining Using Natural Language Processing Neural Network

Background:

User interaction data such as click data from smartphone applications can give useful business insights. However, most of these data do not have process case id.

Pegoraro et al. proposed a novel approach to associate the event data with process case identifier using link-graph (Fig. 3) and word2vec method (Fig. 4) [3]. Word2vec is a natural language processing neural network algorithm originally developed to learn word associations from a corpus of text [4]. The results have been successfully validated to a German Car Sharing company click data.

Objective:

To write a Python code that will implement the method of Pegoraro et al.

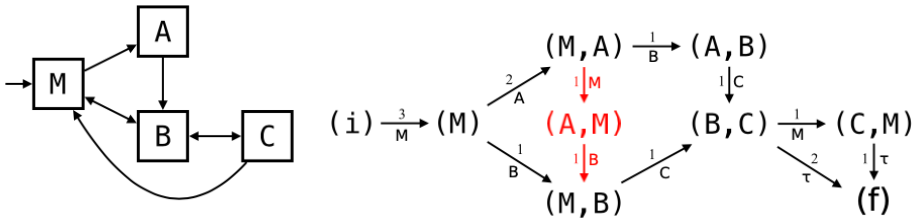

Figure 3: Link graph and resulting transition system.

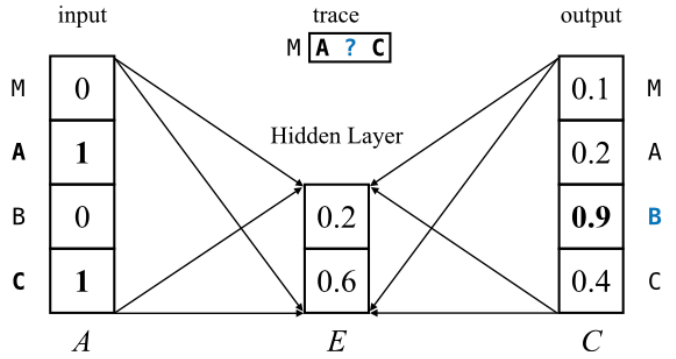

Figure 4: Word2vec neural network for producing probability distribution of activities in between two activities.

[3] Pegoraro et al., (2022) Uncertain Case Identifiers in Process Mining: A User Study of the Event-Case Correlation Problem on Click Data arXiv:2204.04164
[4] Mikolov et al., (2013) Efficient Estimation of Word Representations in Vector Space arXiv:1301.3781

Process Mining Using Natural Language Processing Neural Network

Methods:

Artificial event sequence is constructed by random repetition of the following training traces:

1. A-B-D-E-F
2. A-C-D-E-F
3. A-E-F

A total of 564 training traces has been generated. The training traces has been trained using word2vec model neural network constructed using the open source Pytorch framework [5].

Results:

Fig. 5 shows the loss function during neural netrowk training. Convergence is reached by 200th epoch with a runtime of t = 29.1 sec when run on a 2.2 Ghz single core CPU.

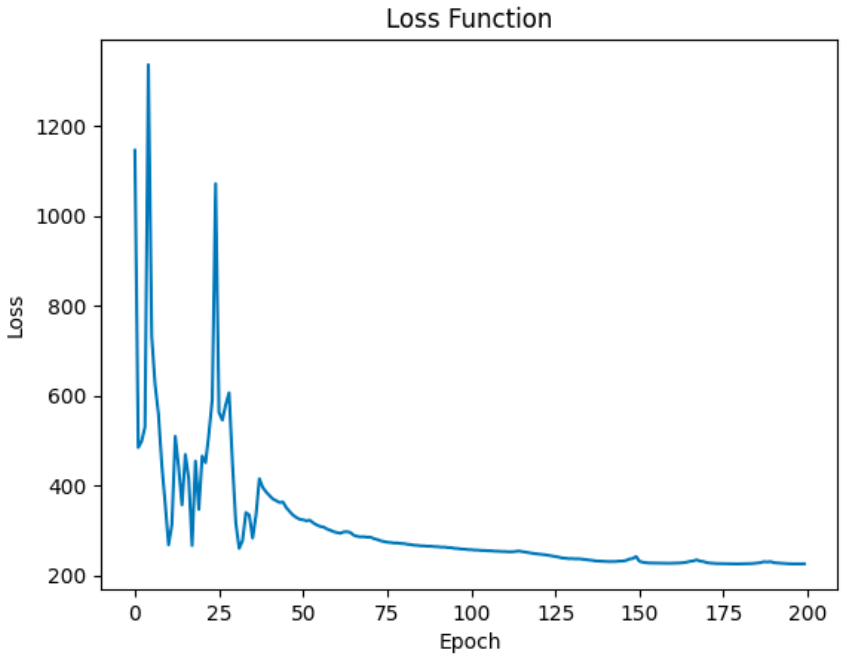

Figure 5: Loss function during neural network traing of training traces.

[5] PyTorch pytorch.org

Process Mining Using Natural Language Processing Neural Network

Results:

The model is tested by predicting the activity in between two activities or context (Fig. 6).

For the first example, the context is [D, F]. The predicted activity is E with probability of 48%, which is much higher than the other candidates. The result is expected since two training traces contain the sequence D-E-F.

For the second example, for the context [A, D], the most probable activity is C (42%) and B (40%). This reflects the training traces that have A-C-D and A-B-D sequence.

```
Training Traces: A-B-D-E-F, A-C-D-E-F, A-E-F
Context:  ['D', 'F']
Prediction:  E
E  =  0.48
x  =  0.17
C  =  0.11
B  =  0.09
A  =  0.09
F  =  0.06
D  =  0.0
```

```
Training Traces: A-B-D-E-F, A-C-D-E-F, A-E-F
Context:  ['A', 'D']
Prediction:  C
C  =  0.42
B  =  0.4
F  =  0.07
D  =  0.04
E  =  0.04
A  =  0.02
x  =  0.0
```

Figure 6: Activity prediction given the context activity.

Process Mining Using Natural Language Processing Neural Network

Results:

Finally, the model is used to segment the symbol sequence. Pegoraro et al. proposed that *case boundary* is likely to happen if the probability score of the predicted activity p_i is higher than the previous activities multiplied to tunable hyperparameters b_1, b_2, b_3, and k.

$$\text{If } p_i > b_1 p_{i-1} \text{ and,}$$

$$\text{If } p_i > b_2 p_{i-2} \text{ and,}$$

$$\text{If } p_i > b_3 \frac{\sum_{j=i-k-1}^{i-1} p_j}{k}$$

Then p_i is a case boundary.

A segment of the probability scores of the predicted activity is plotted in Fig. 7, where we can see two peaks (A and E). The hyperparameters can be set to distinguish case A from E. The case boundaries will be used to split the symbol sequence.

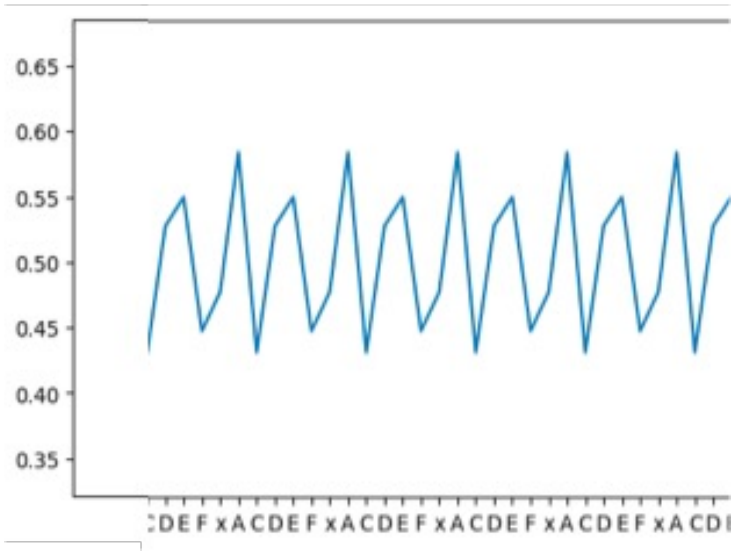

Figure 7: Plot of the predicted cases and their probability scores.

Process Mining Using Natural Language Processing Neural Network

Results:

We can see from Fig. 8 that the model has been successful in splitting the events [A-B-D-E-F] and [A-C-D-E-F]. However, it finds it difficult to split cases adjacent to [A-E-F]

Conclusions:

A process mining program based on the work of Pegoraro et al. has been written using Pytorch neural network framework. The general features of the original work has been reproduced. The model gave some incorrect predictions. Whether this error is due to the *limitations of the program* or due to the *artifically produced training data* is yet to be determined.

Recommendations:

Test the model in other training data and to develop a more rigorous method of optimizing the hyperparameters.

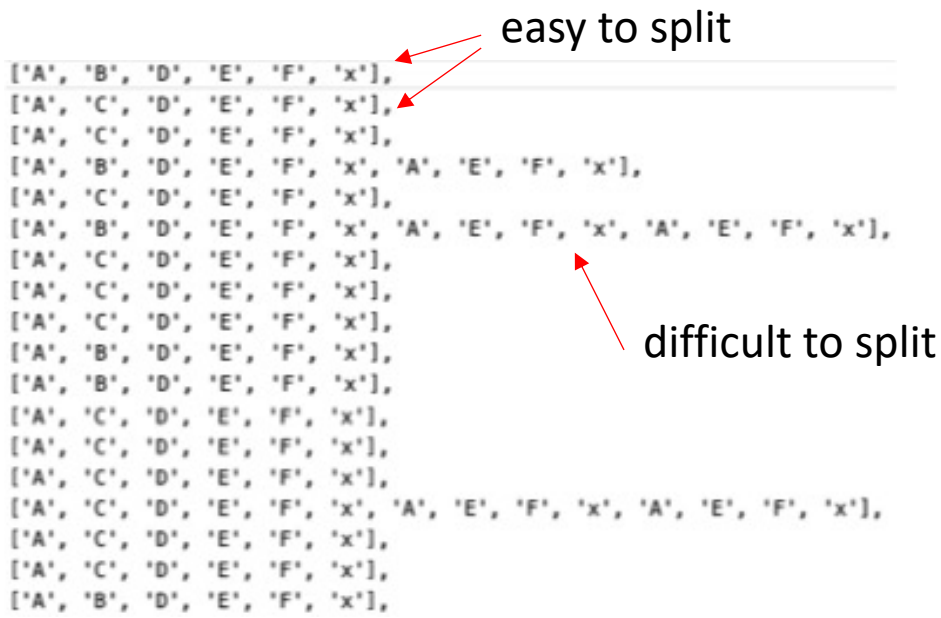

Figure 8: Splitting the events using the predicted case boundaries.

Project 3
Process Mining Using Population Based Simulated Annealing

Process Mining Using Population Based Simulated Annealing

Background:

Simulated annealing is an optimization method that searches for the nearest approximate global solution by simulating the *cooling process* of metals through the annealing process [6]. The results found by simulated annealing depends on the initial solution. When the initial solution is not good, it can get trapped in a local minima.

Population based simulated annealing uses several initial solutions (populations) that have shared memory of past experiences. The sharing of "elite" experiences between populations allows them to avoid local minima and reach the global solution [7].

Recently, population based simulated annealing has been applied in process mining of unlabeled event logs [8].

Objective:

To write a Python code that will implement the population based simulated anneling and process mining using population based simulated annealing.

[6] Kirkpatrick et al. (1983) Optimization by simulated annealing. Science **220** (4598): 671-680.
[7] Askarzadeh et al., (2016) A Population-Based Simulated Annealing Algorithm for Global Optimization SMC. IEEE 2016
[8] Bayomie et al. (2019)A Probabilistic Approach to Event-Case Correlation for Process MiningConceptual Modeling pp 136–152

Process Mining Using Population Based Simulated Annealing

Methods:

Simulated annealing starts by randomly choosing an initial solution x and calculating its fitness $f(x)$. A candidate solution x' with fitness $f(x')$ is selected around the vicinity of the initial solution. The candidate solution is chosen based on the following criteria:

$$\text{Criteria 1: } \Delta F = f(x') - f(x) \leq 0, \text{ or}$$

$$\text{Criteria 2: } \exp\left(\frac{-\Delta F}{T}\right) > r, \text{ where } T = \frac{T_0}{\ln(1+t)}$$

Here, T is the annealing temperature which depends on the iteration number t and r is a random number. From the first criteria, if the candidate solution is better than the previous (ΔF is negative), then it is automatically selected. However, even if the candidate solution is not better, there is a chance that it will be selected based on the second criteria. In population based simulated annealing, the generation of candidate solution for ith population x_i' is affected by past "elite" or good solutions of every other population $j \neq i$.

$$x_i' = x_i + w \times \left[r_1\left(x_j^{elite1} - x_i\right) + r_2\left(x_j^{elite2} - x_i\right)\right]$$

where w is a time-varying parameter and r_1 and r_2 are random numbers.

Process Mining Using Population Based Simulated Annealing

Results:

Function: $f(x) = x^2$
True minimum: $f(x = 0) = 0$

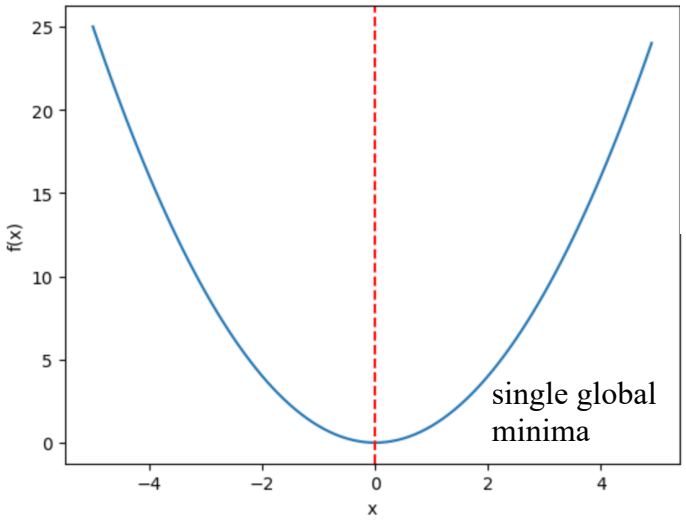

Figure 8: $f(x) = x^2$

Population based simulated annealing **reach the solution faster**.

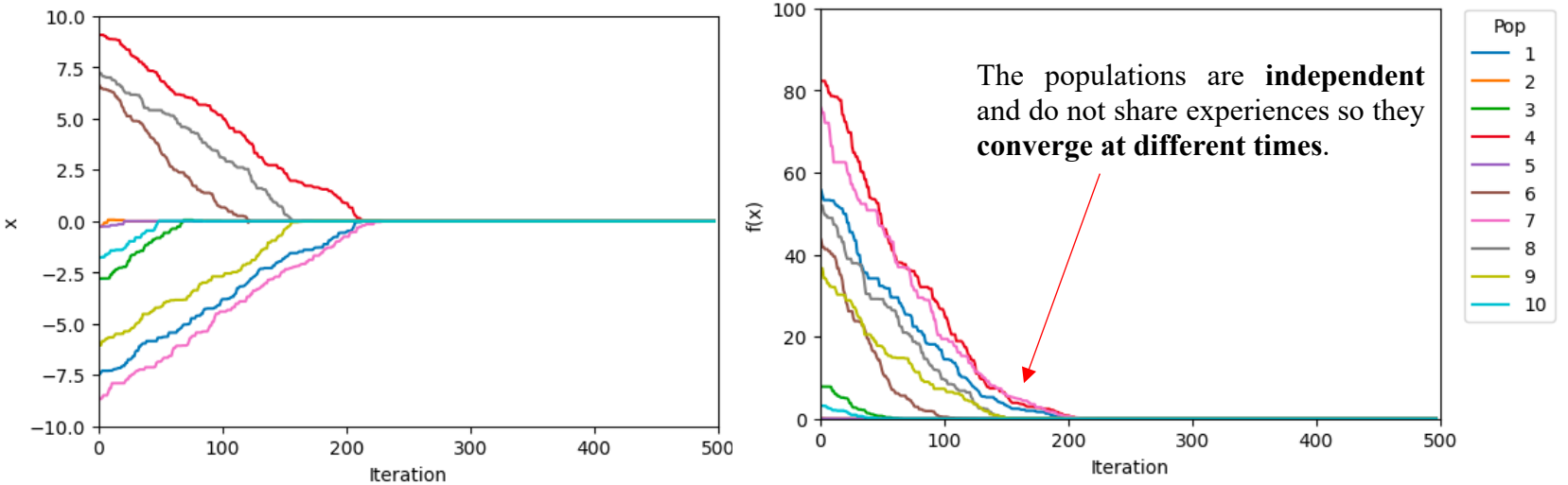

Figure 9: Simulated anneling (populations are independent)

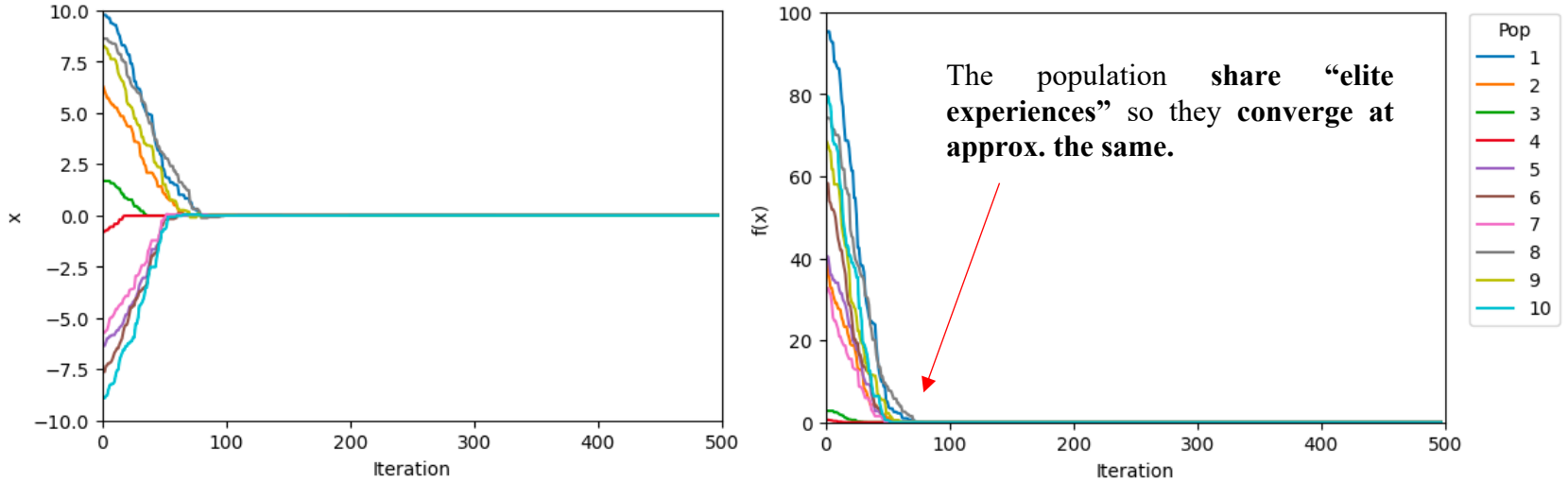

Figure 10: Population Based Simulated anneling (populations share "elite experiences")

Process Mining Using Population Based Simulated Annealing

Results:

Function: $f(x) = (x + rand)^2$
True minimum: $f(x = 0) = 0$

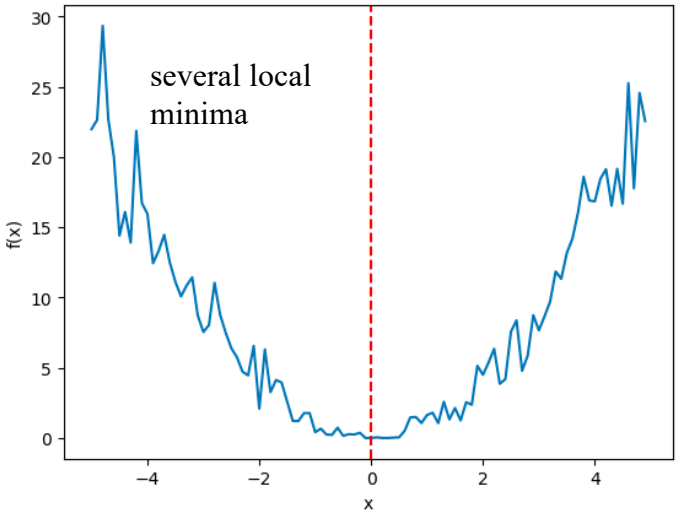

Figure 11: $f(x) = (x + rand)^2$

Population based simulated annealing **prevented local minima solutions**.

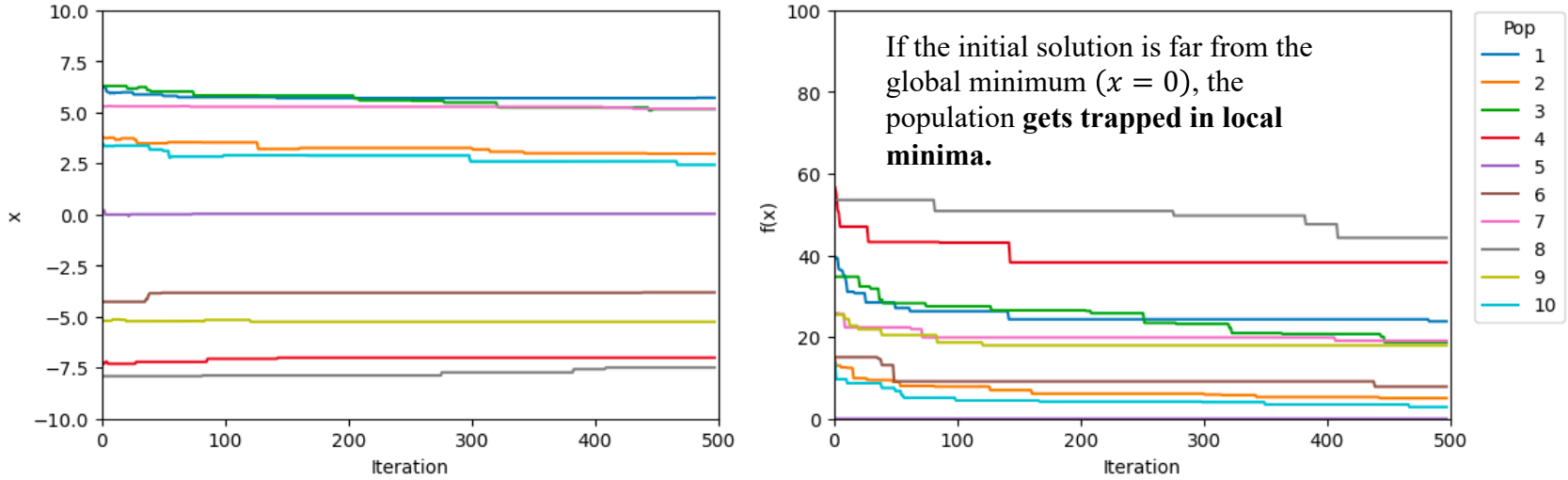

Figure 12: Simulated anneling (populations are independent)

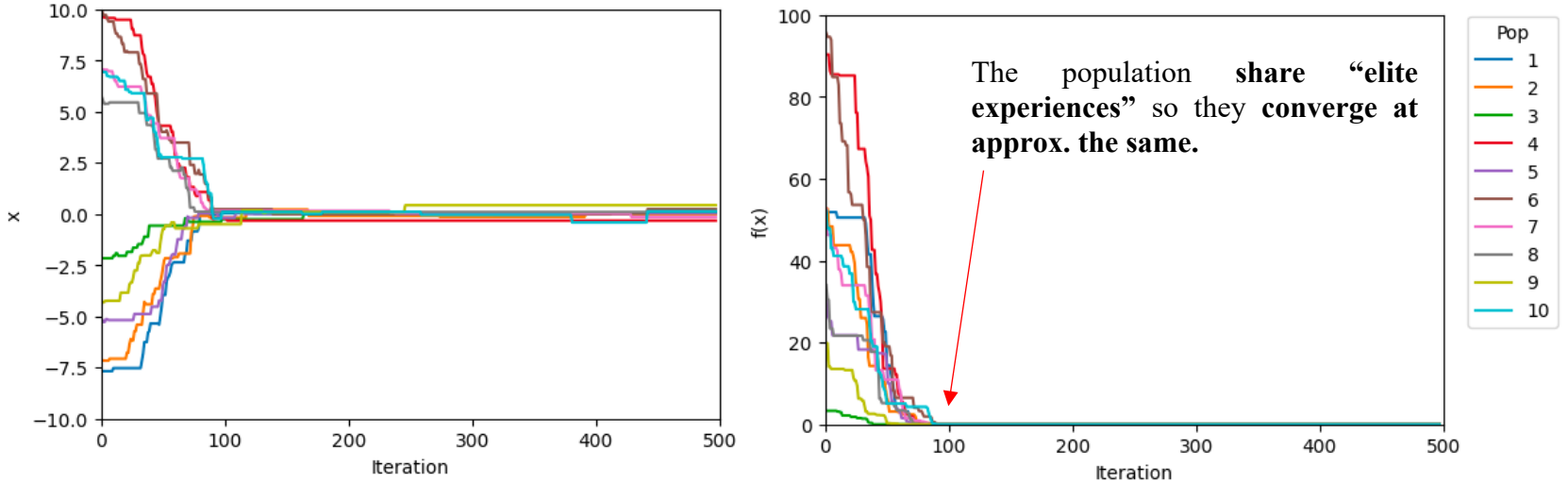

Figure 13: Population Based Simulated anneling (populations share "elite experiences")

Process Mining Using Population Based Simulated Annealing

```
Iter 2
Current Case ID :  [1, 2, 1, 2, 2, 1, 3, 3] fa =  0.0 ft =  4.062
Candidate Case ID :  [1, 2, 2, 2, 2, 1, 3, 3] fa_candidate =  2.0 ft_candidate =  4.062
Candidate Case ID Accepted Randomly
Current Case ID :  [1, 2, 2, 2, 2, 1, 3, 3] fa =  2.0 ft =  4.062
Iter 3
Current Case ID :  [1, 2, 2, 2, 2, 1, 3, 3] fa =  2.0 ft =  4.062
Candidate Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa_candidate =  0.0 ft_candidate =  0.3
Candidate Case ID is Accepted since fa_candidate < fa
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Iter 4
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Candidate Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa_candidate =  0.0 ft_candidate =  0.3
Candidate Case ID is not Accepted
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Iter 5
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Candidate Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa_candidate =  0.0 ft_candidate =  0.3
Candidate Case ID is not Accepted
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Iter 6
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Candidate Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa_candidate =  0.0 ft_candidate =  0.3
Candidate Case ID is not Accepted
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Iter 7
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Candidate Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa_candidate =  0.0 ft_candidate =  0.3
Candidate Case ID is not Accepted
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Iter 8
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Candidate Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa_candidate =  0.0 ft_candidate =  0.3
Candidate Case ID is not Accepted
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Iter 9
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
Candidate Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa_candidate =  0.0 ft_candidate =  0.3
Candidate Case ID is not Accepted
Current Case ID :  [1, 2, 2, 2, 1, 2, 3, 3] fa =  0.0 ft =  0.3
```

Figure 14: Optimization of Case ID Fitness Scores using Simulated Anneling

Simulated Annealing has been successfully implemented to optimize the fitness scores of candidate Case IDs.

Process Mining Using Population Based Simulated Annealing

Conclusions:

Population based simulated annealing performs better than the standard simulated annealing algorithm.

Simulated Annealing has been successfully implemented to optimize the fitness scores of candidate Case IDs.

Recommendations:

Test the effectiveness of population based simulated annealing in process mining of unlabeled event logs in real-world data.