Technical Report
"Cut to the Trace!"

Gaël Bernard, Arik Senderovich, and Periklis Andritsos

University of Toronto, Faculty of Information, Toronto, Canada
gael.bernard@utoronto.ca
arik.senderovich@utoronto.ca
periklis.andritsos@utoronto.ca

This technical report extends the customer journey partitioning experiments from our publication:

G. Bernard, A. Senderovich, and P. Andritsos, "Cut to the trace! process-aware partitioning of long-running cases in customer journey logs," in *Advanced Information Systems Engineering*, M. La Rosa, S. Sadiq, and E. Teniente, Eds. Cham: Springer International Publishing, 2021, pp. 519–535

The report is composed of three parts. First, we test the performance of the partitioning techniques with various event logs. Second, we measure the scalability of the approaches using event logs of varying sizes. Third, we conclude the report. The implementation of the three methods, the experiment from the original publication, and the ones from this technical report are available online[1].

1 Performance Analysis

In our publication, we generated 10 customer journeys by stacking 100 traces generated from a known process model. The experiment's goal was to measure the partitioning techniques' ability to retrieve the original 100 stacked traces of each journey without prior knowledge of the process model used to generate the journeys. We simulated the time between events using a Poisson process and added a delay when two events belong to two distinct process instances. The smaller the delay, the harder it is to retrieve the original partitioning.

In this technical report, we extend this experiment by using 10 process models defined in Leemans' thesis [2]. From these process models, Leemans produced event logs containing 1024 traces[2]. The key characteristics of the 10 process models are described in Table 1. We also depicted the first process model, seed 1, in Fig. 1.

Table 1. Key characteristics of the 10 process models.

Seed	**#Activities**	**#Xor**	**#Parallel**	**#Sequential**	**#Loops**	**Simplicity [3]**
1	32	5	4	4	1	0.642
2	32	6	3	6	2	0.670
3	32	6	3	7	3	0.681
4	32	0	3	5	3	0.811
5	32	2	4	7	0	0.659
6	32	3	5	4	0	0.670
7	32	4	5	3	1	0.630
8	32	1	5	10	3	0.786
9	32	5	5	5	2	0.686
10	32	0	5	7	3	0.756

[1] https://github.com/gaelbernard/cjp
[2] Available at: https://data.4tu.nl/articles/dataset/A_collection_of_artificial_event_logs_to_test_process_discovery_and_conformance_checking_techniques/12704777 under '1 - scalability/generatedLogs/round 5 treeSeed *.xes.gz'

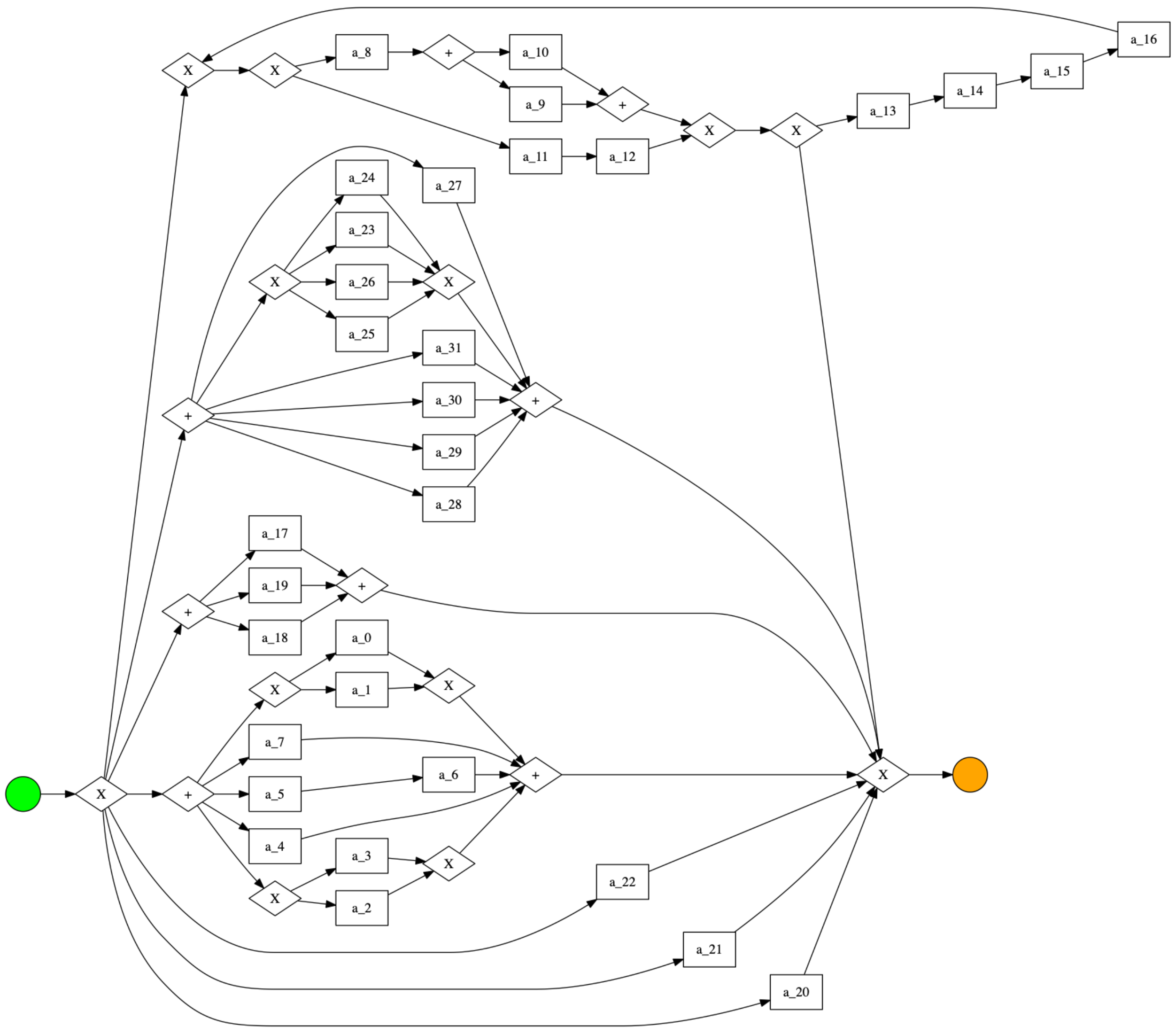

Fig. 1. One of the ten process models, seed 1, used to generate the event logs.

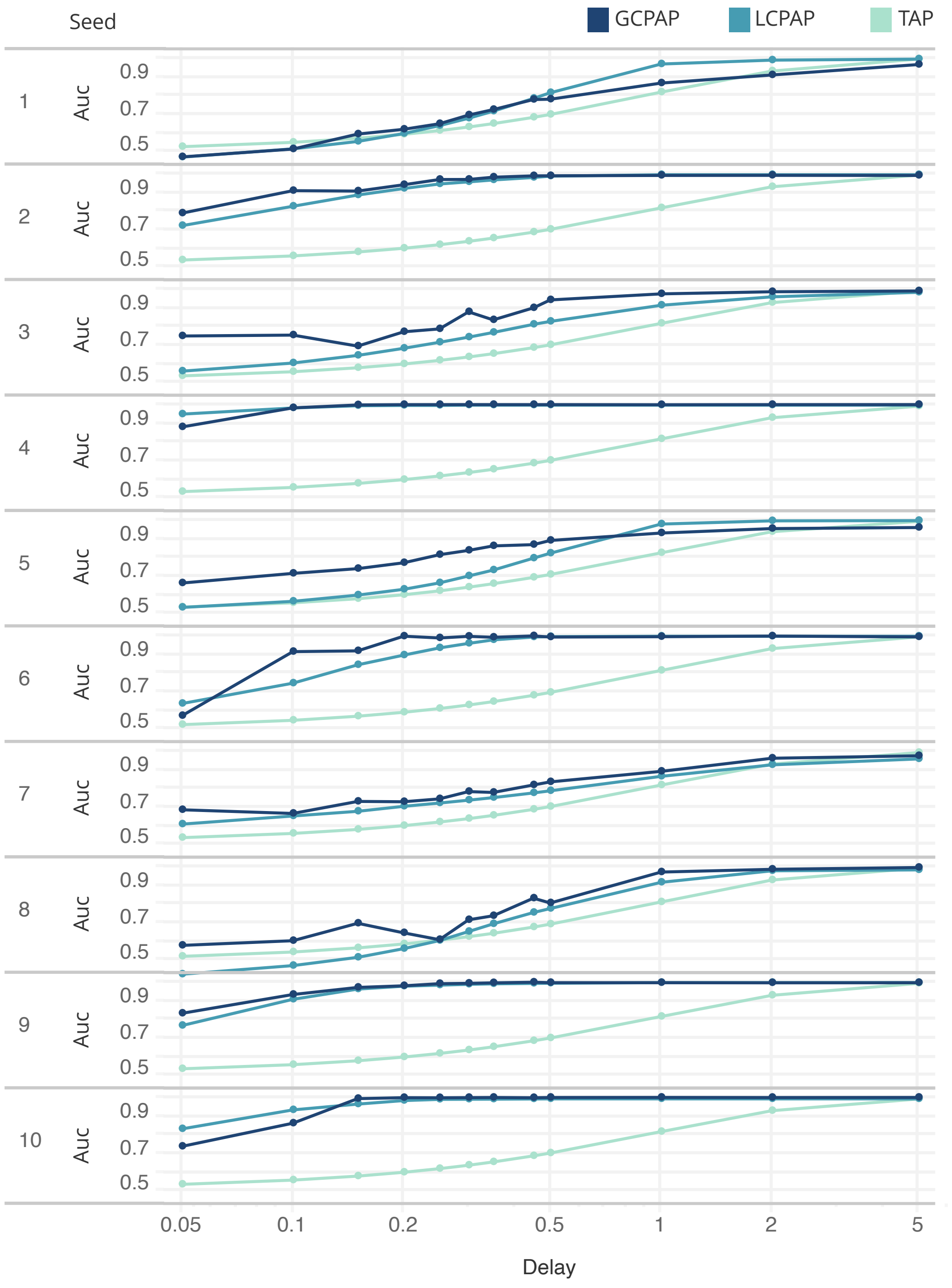

Fig. 2. AUCs of the three approaches

We borrowed these 10 event logs that we transformed into customer journey logs by stacking traces separated with 12 fixed delays and a Poisson distribution with a rate, λ, fixed to 1 as described in the original paper [1]. Altogether, we produced 120 customer journey logs that we partitioned using TAP (Time Aware Partitioning), GCPAP (Global Context Process-Aware Partitioning), and LCPAP (Local Context Process-Aware Partitioning).

The results in terms of Area under the Curve (AUC) are shown in Fig. 2. On 96 occasions, GCPAP outperforms the two other approaches, while LCPAP gets the best AUC 20 times. There are 4 instances where TAP achieved the highest AUC.

If the delay between inter-case times is almost constantly higher than the inter-event times, using more sophisticated partitioning methods is overkill, which is well represented with a delay of 5 in Fig. 2. In fact, when the Poisson distribution rate is set to 1 and the delay is 5, the probability that an inter-event time becomes larger than the delay is only 0.31% ($f(d, \lambda) = \frac{\lambda^x e^{-\lambda}}{d!}$), which explains the good performance of the three techniques under this particular delay.

Interestingly, we do not see any correlation between the performance manifested by the AUC and the process model's complexity. For instance, looking at Fig. 2, the partitioning of journeys is more challenging in seed 8 since the AUCs are lower, while seed 8 is one of the most straightforward process models according to metric borrowed from [3].

In Fig. 3, we plotted the ROC curves for the most challenging delay, i.e., when it is set to 0.05. With such a small delay, TAP always performs a bit better than a random guest. On rare occasions, GCPAP and LCPAP perform worst than a random guest, e.g., seed 1. However, most of the time, these approaches outperform a random guest and TAP.

Interestingly, in seed 6, GCPAP and LCPAP achieve spectacular AUCs of, respectively, 0.88 and 0.95. We explain this because, for this process model, the traces always start and end with the same activities. In this case, a trace partitioning technique is typically not required. It is the only process model that enforces such static activities at the beginning or the end of the trace. Still, we highlight that these excellent AUCs were obtained without having any information about the underlying process model.

In the next section, we discuss the scalability of the three partitioning techniques.

2 Scalability

To measure the three approaches' scalability, we leverage the same collection of datasets from [2]. In the previous section, we used round 5. Altogether, there are 10 rounds of increasing complexity and size, each containing 10 event logs. Due to time constraints, we limited the scalability test to the first seventh rounds. We described them in Table 2.

Table 2. Characteristics of the event logs used for the scalability evaluation.

Round	**#Event Logs**	**#Activities**	**#Cases**	**Avg. #events**
		per event logs	per event logs	per event logs
1	10	2	4	6.8
2	10	4	16	60.4
3	10	8	64	501.8
4	10	16	256	3'282.8
5	10	32	1'024	26'316.0
6	10	64	4'096	154'509.2
7	10	128	16'384	1'387'570.0

The machine used for the experiment has 16GB of RAM, and 4 CPUs. We did not use GPU, which could speed up the neural network-based partitioning, i.e., GCPAP. The parameters for the GCPAP remain identical to the ones used in the paper, i.e., w to 10 (size of window activities), n to 64 (number of units in LSTM and dense layers), and the number of epochs to 20.

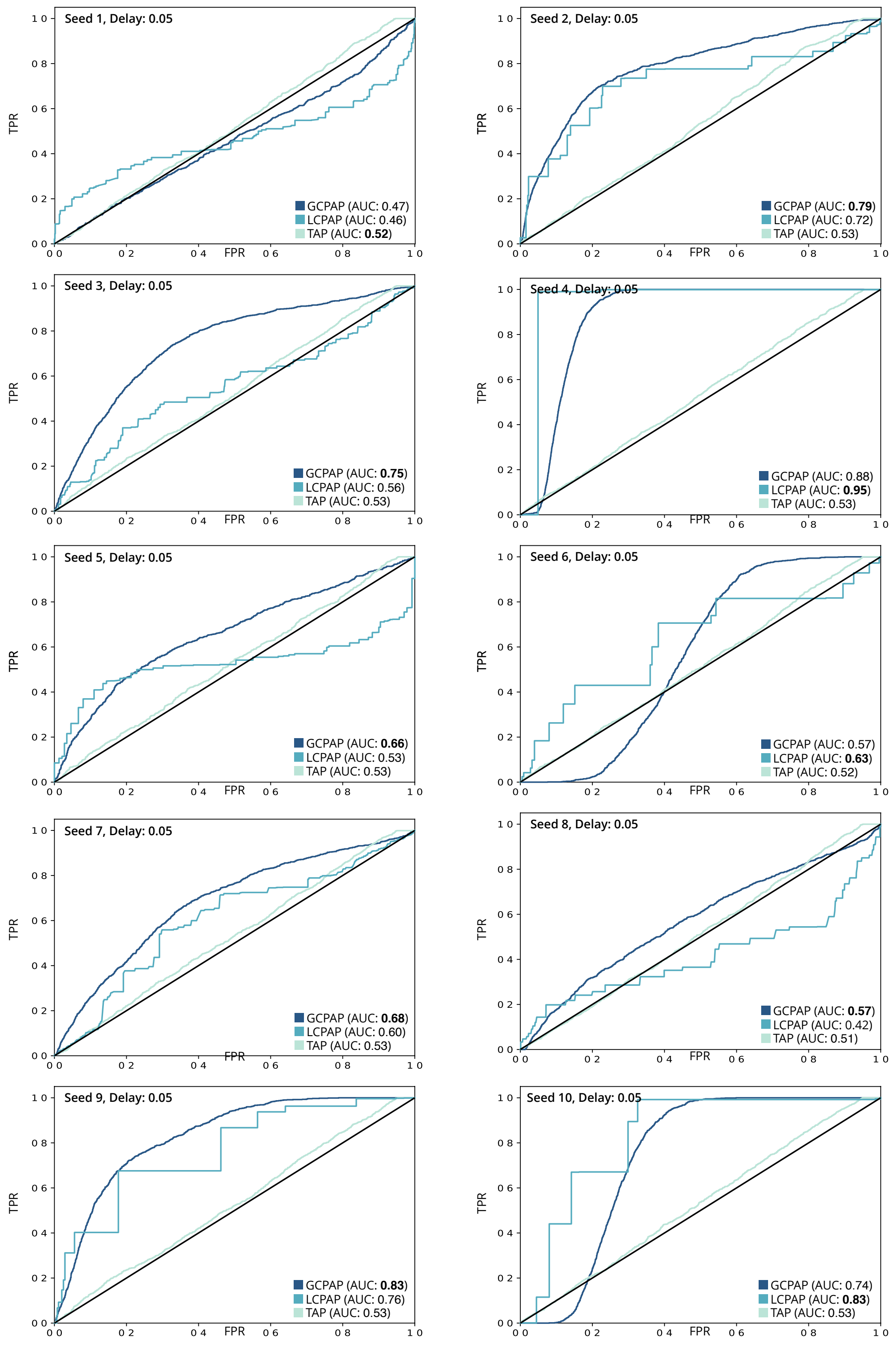

Fig. 3. ROC curves for the delay 0.05

The scalability test is visible in Fig. 4. Note the y-axis is using a log scale. Since TAP only consists of looking for the largest time delta between events, it is not surprising that this approach can operate within a fraction of seconds, even for heavy event logs. What can be noted is that the LCPAP approach can also return partition in a reasonable amount of time. For instance, for the largest event logs, which contain 2.02M events, the runtime of LCPAP is 5.5 seconds. The runtime of GCPAP is clearly much slower by a factor of 700: it took a little bit over an hour (3850 seconds) for the same dataset. Unlike TAP and LCPAP, one can arguably state that such a long run time does not allow for online exploration of event logs. However, regarding the journeys partitioning improvement highlighted in the first section of this technical report, we believe that LCPAP has considerable potentials worth the run time.

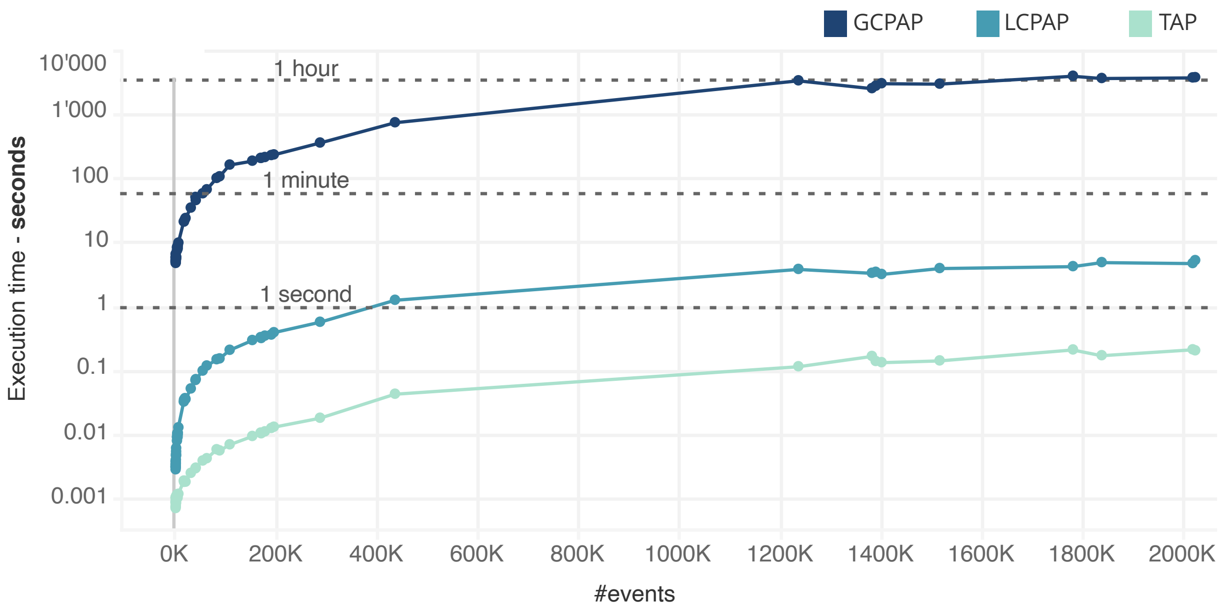

Fig. 4. Execution time of the three approaches

3 Conclusion

We have shown in this report that the excellent performance obtained by our two novel approaches for partitioning a customer journey, namely LCPAP and GCPAP, stands for various and independent synthetic event logs. We have also shown that the execution time overhead of LCPAP compared to TAP is limited, while it is quite significant for GCPAP.

References

1. G. Bernard, A. Senderovich, and P. Andritsos, “Cut to the trace! process-aware partitioning of long-running cases in customer journey logs,” in *Advanced Information Systems Engineering*, M. La Rosa, S. Sadiq, and E. Teniente, Eds. Cham: Springer International Publishing, 2021, pp. 519–535.
2. S. Leemans, “Robust process mining with guarantees,” Ph.D. dissertation, Mathematics and Computer Science, May 2017, proefschrift.
3. 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,” *International Journal of Cooperative Information Systems*, vol. 23, no. 01, p. 1440001, 2014.