Stuff for the thesis, but too bloated for a paper.

Ignore this visual. It seems unnecessary for the paper, since the D1/D2 results for the Sampling algorithm were nearly identical. But the results should be included in the thesis, so I’m leaving them prepared here.

|  |  |
| --- | --- |
| XX.1 | XX.2 |
| XX.3 | XX.4 |
| Figure-set XX.x: From top-left, dataset D2 Sample Algorithm results, accuracy (1), f1-measure (2), recall (3), and precision (4). The results show nearly identical Sampling Algorithm performance on both D1 and D2. | |

Noise Generation [obsolete?]

The model and trace generation methods are sufficient to generate spaghetti-like process data in terms of model complexity via θ\_model and activity complexity via θ\_traces, but remain constrained to the given model. For example, under this data generation method there will be no log ambiguity as to whether an activity occurs before another if it always lies upstream in the paths contained in the model; hence, the underlying model strongly constrains the generated data. Although likely unusual, one might expect unstructured work environments to inevitably lead to occasional repetitions, whereby sub-components of some model are repeated or activities do not always follow one another. Such a scenario occurs, for instance, if prior to some software release a bug is detected after final testing, causing components of the testing activities to be repeated arbitrarily with respect to the software development process.

The simplest way to simulate worst-case noise is to randomly inject activities into the traces after trace generation. For every activity in each trace, with probability α a random activity is inserted. The chosen activity is selected at uniform random from the set of all activities defined over the log. This method of noise addition is necessary to test our approach’s resilience to noise, and is justified since it creates many data outliers that would be too easily declared anomalous by an anomaly detection method that does not distinguish outliers from anomalies. Note that even small values of α tend to create highly obfuscated partially-orderings for a process mining algorithm to mine: for a log of 1000 traces, for which the average trace length is 25, an α value of 0.1 will have an expected number of “noise” activity occurrences of 0.1 \* 25 \* 1000 = 2500. This is an extreme number of random activities, since even a single misplaced activity with respect to the underlying model, breaks the logical rules and heuristics applied by process mining algorithms to mine the regular features of the underlying process model.

Raw plottable data values (of 2566 traces):f for bayes xs [0.01, 0.03, 0.05, 0.07, 0.09] ys were [18,183,966,1348,1620].