KU LEUVEN

Enhancing Stochastic Petri Net-based Remaining Time Prediction using k-Nearest Neighbors

Jarne Vandenabeele, Gilles Vermaut

Jari Peeperkorn, Jochen De Weerdt

RESEARCH CENTRE FOR INFORMATION SYSTEMS ENGINEERING (LIRIS)

KU Leuven, Belgium

Research Center for Information Systems Engineering (LIRIS), Faculty of Economics and Business

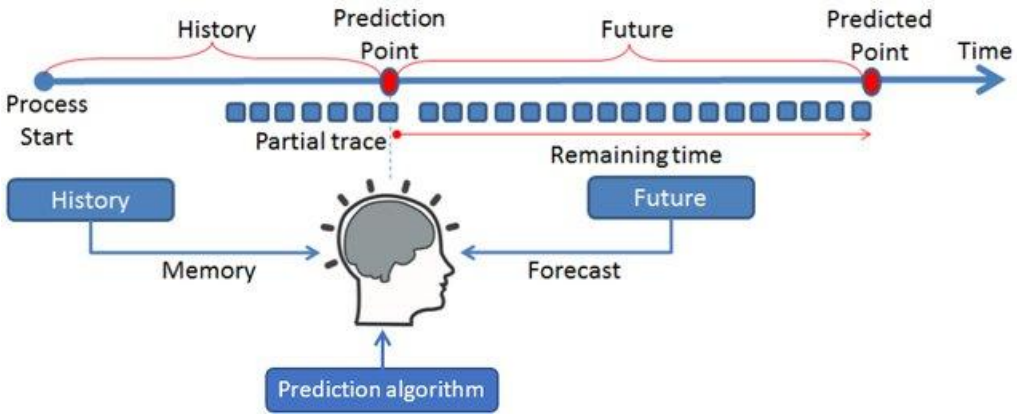
Contents

Introduction

Introduction

- Predictive process monitoring
  - Remaining time prediction

- Multiple approaches
  - Regression based approaches
  - Transition systems
  - …
  - Machine Learning
    - Random Forests, XGBoost, Neural Networks…



From Verenich et al.(2019).

- Generally distributed transition stochastic Petri net (GDT_SPN)
  - Rogge Solti and Weske (2013, 2015)

- Combined with a simple, yet effective, data-driven candidate selection using kNN

Method

Method

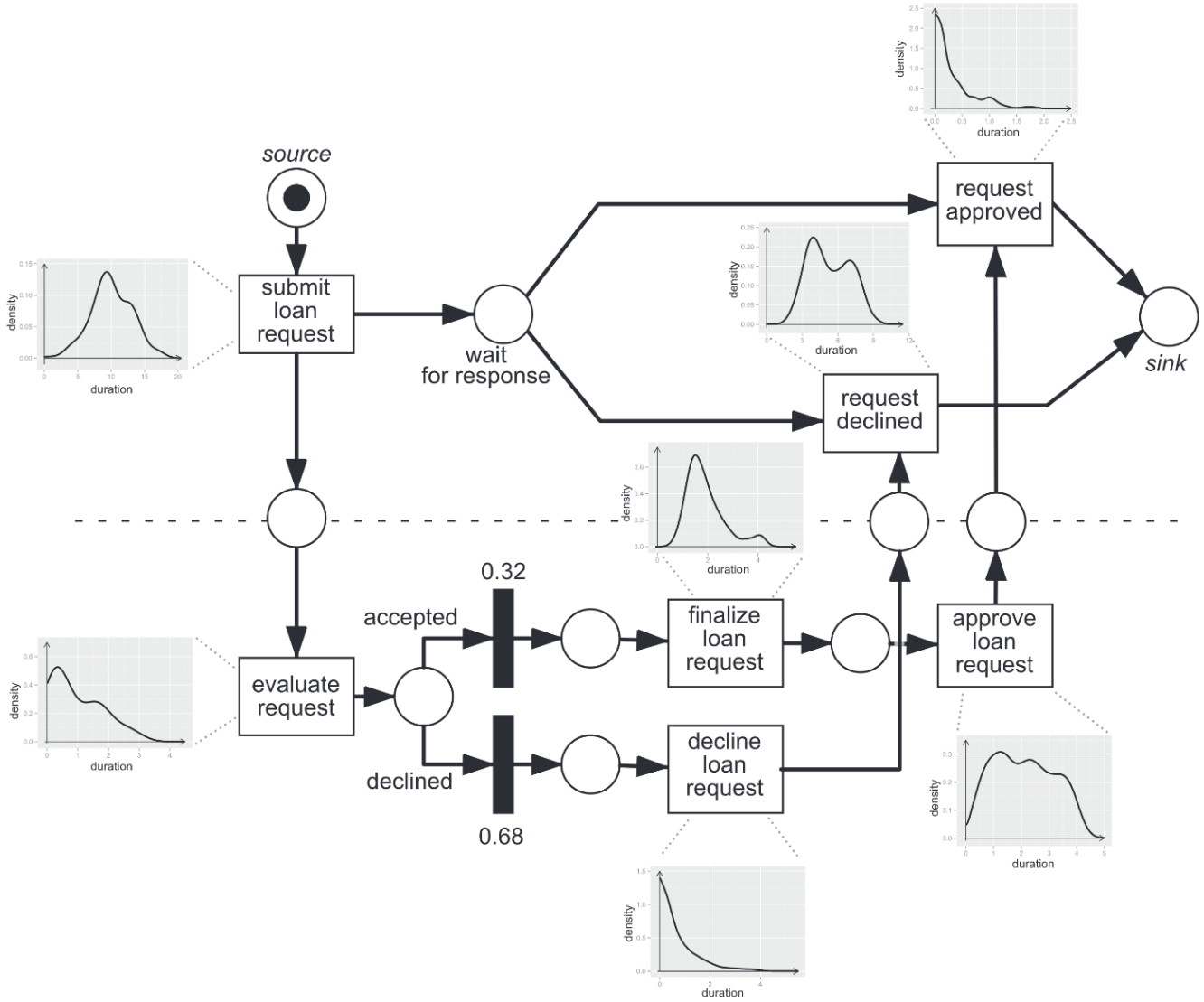
- Use the K-nearest-neighbor algorithm to identify the k instances most similar to the current prefix of this instance. The algorithm uses vectors based on the timestamps of previous events, while also taking into account the activity types.

- We use these k traces to discover a (new) Petri net and by performing a simulation a stochastic map is obtained which complements the Petri net, to eventually obtain a GDT_SPN.

- A simulation of this GDT_SPN is further used to estimate the remaining time. This is done n times and the actual prediction is taken to be the average of the n simulations

Generally Distributed Transition Stochastic Petri Net

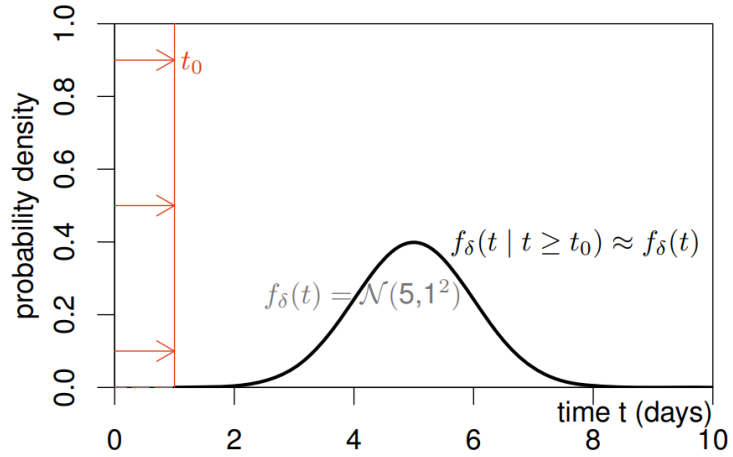
- A basic Petri Net (set of places, set of transitions, a flow relation and an initial marking)
- Immediate and timed transitions
- Probabilistic weights to immediate transitions
- Arbitrary probability distributions to the timed transitions (duration activity)
- Priorities that ensure that if immediate transitions are enabled, no timed transition can fire



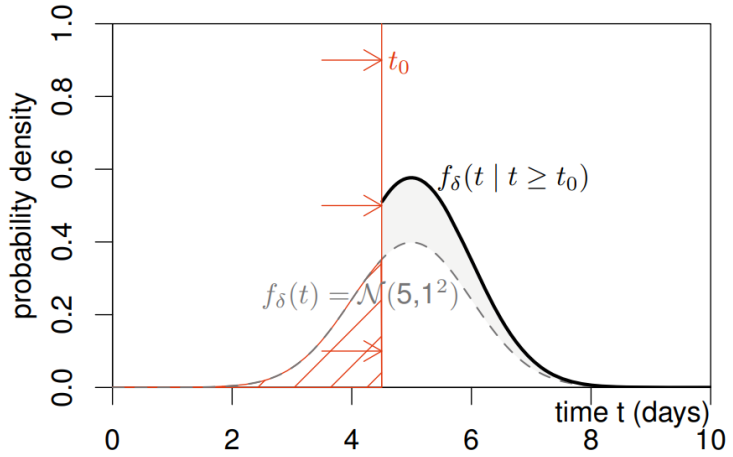
Taken from (Rogge-Solti, M. Weske, 2015)

Prediction using a GDT_SPN

- Prediction of a current ongoing case:
  - Replay the (incomplete) trace with the model
  - Simulate the rest of the case
    - Use a truncated density for each activity
  - *n* iterations - average



(a) After one day, the truncated density is almost unchanged



(b) After 4.5 days, the truncated density is significantly different from the original density

Taken from (Rogge-Solti, M. Weske, 2013)

K-nearest neighbors

- We do not use all available traces to build the GDT_SPN
  - Only the k cases which are most similar (up to that point in the trace)

- You'll need a way to measure the distance between two (partial) cases
  - Flexible, could be chosen based on expert knowledge

- This is a lazy learner
  - Does not learn during training, but select at runtime
  - More flexible (changing process)
  - Less efficient when deployed

Distance

- Opted for a simple first attempt
  - Not taking into account different event features
  - Applicable to most event logs

- *Time-to-occurrence* of each activity type
  - Time since start case
  - Vector dimension = size activity voc.
  - In case of looping/repetition use last occurrence
  - If a certain activity type does not occur we give a value of -1

- When finding neighbors
  - Only use positive dimensions of trace in question (only occurred activities)
  - Min-max normalization (to give all activities same influence)
  - Euclidian distance

Parameters

- The number of neighbors
  - Set to $k = 100$ (empirically set, no extensive experimentation)
  - Too high = loss of reason to take a subset + computation time
  - Too low = too volatile

- The number of simulation iterations n by GDT_SPN
  - $n = 500$

Code

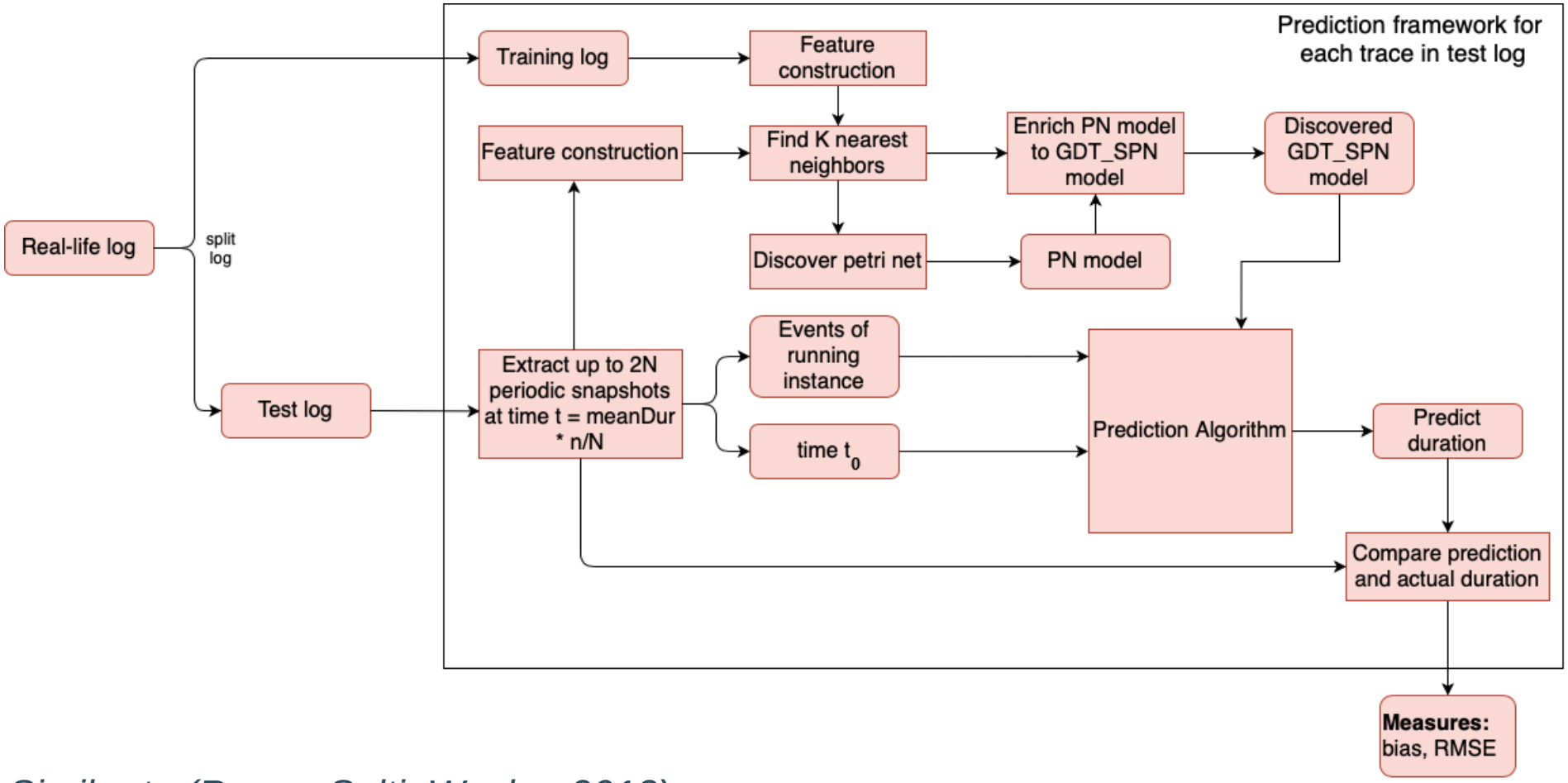
- Available online[1]
- Python
  - Pm4py[2]

[1] *https://github.com/JarneVDB/BP-Time-Prediction-using-KNN*

[2] *https://pm4py.fit.fraunhofer.de/*

Experimental Evaluation

Prediction framework



Similar to (Rogge-Solti, Weske, 2013)

Prediction framework

- 2*N* periodic prediction iterations performed (for each trace)

- *N* = parameter that indicates how many different predictions we want to make
- The *N*th prediction will be performed after the mean duration of all cases

- We set *N*=20, so we have 40 points in time (since start specific case) where we make a prediction about the remaining time (equally spaced)
- Each time we provide the algorithm with the current trace upon to that point in time (= prediction iteration)
- If a case is done, it will not influence the iterations occurring after this point

Metrics

- RMSE
  - Accuracy method
- Average Error
  - Bias

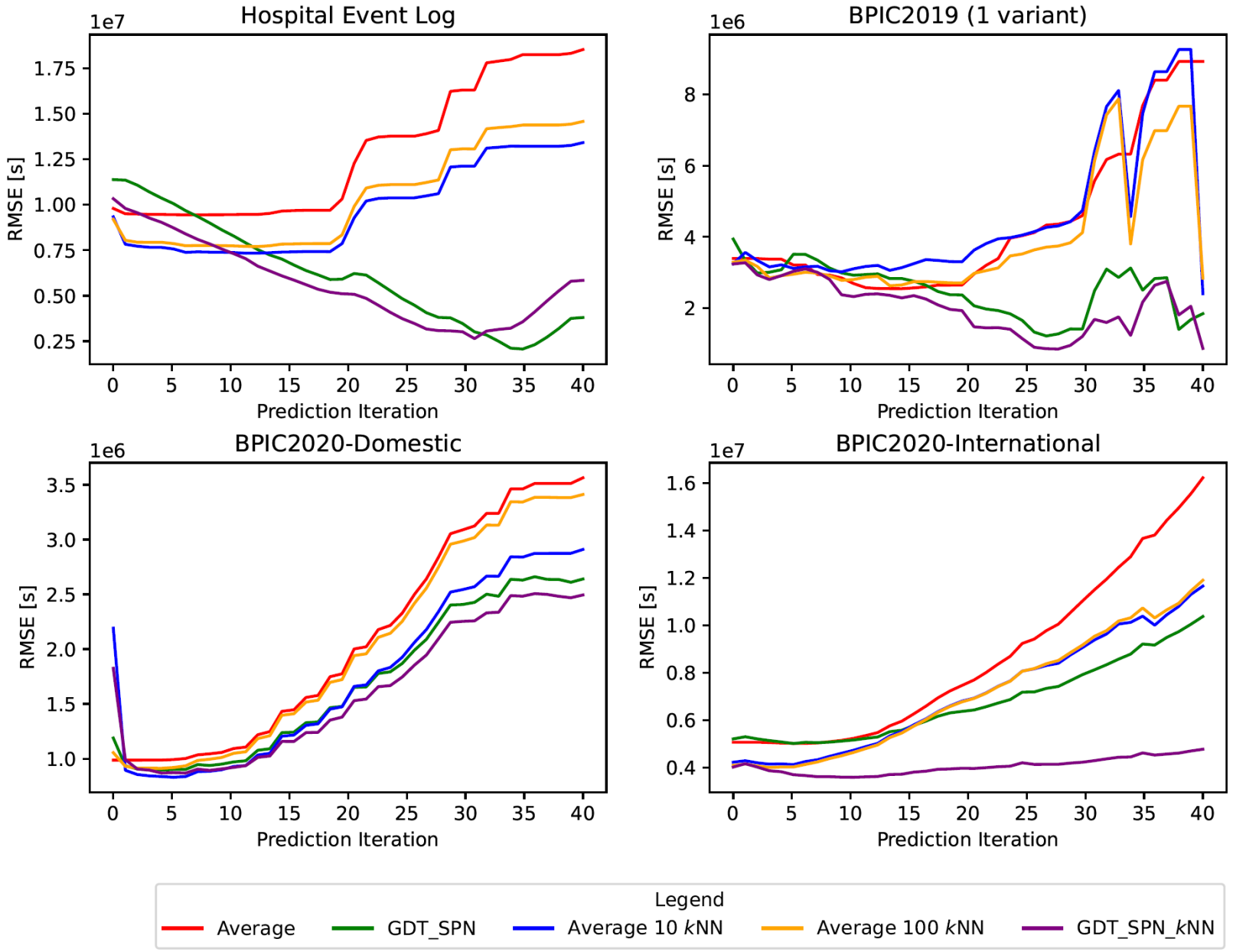
Event Logs

Datasets	Cases	Events	Event classes	Max case length	Avg. case length	Max case time	Avg. case time
BPIC2019 (1 variant)	7460	44760	6	6	6	356.21	94.54
Hospital Billing	7847	33450	7	6	4.26	867.54	156.95
BPIC 2020 (domestic)	7820	40281	7	6	5.15	290.89	10.52
BPIC 2020 (international)	2361	23726	14	12	10.05	463.04	80.17

- 500 cases test set
- Split out-of-time: the training log used to obtain the k-nearest neighbors uses only traces finished upon that point in time

Comparison

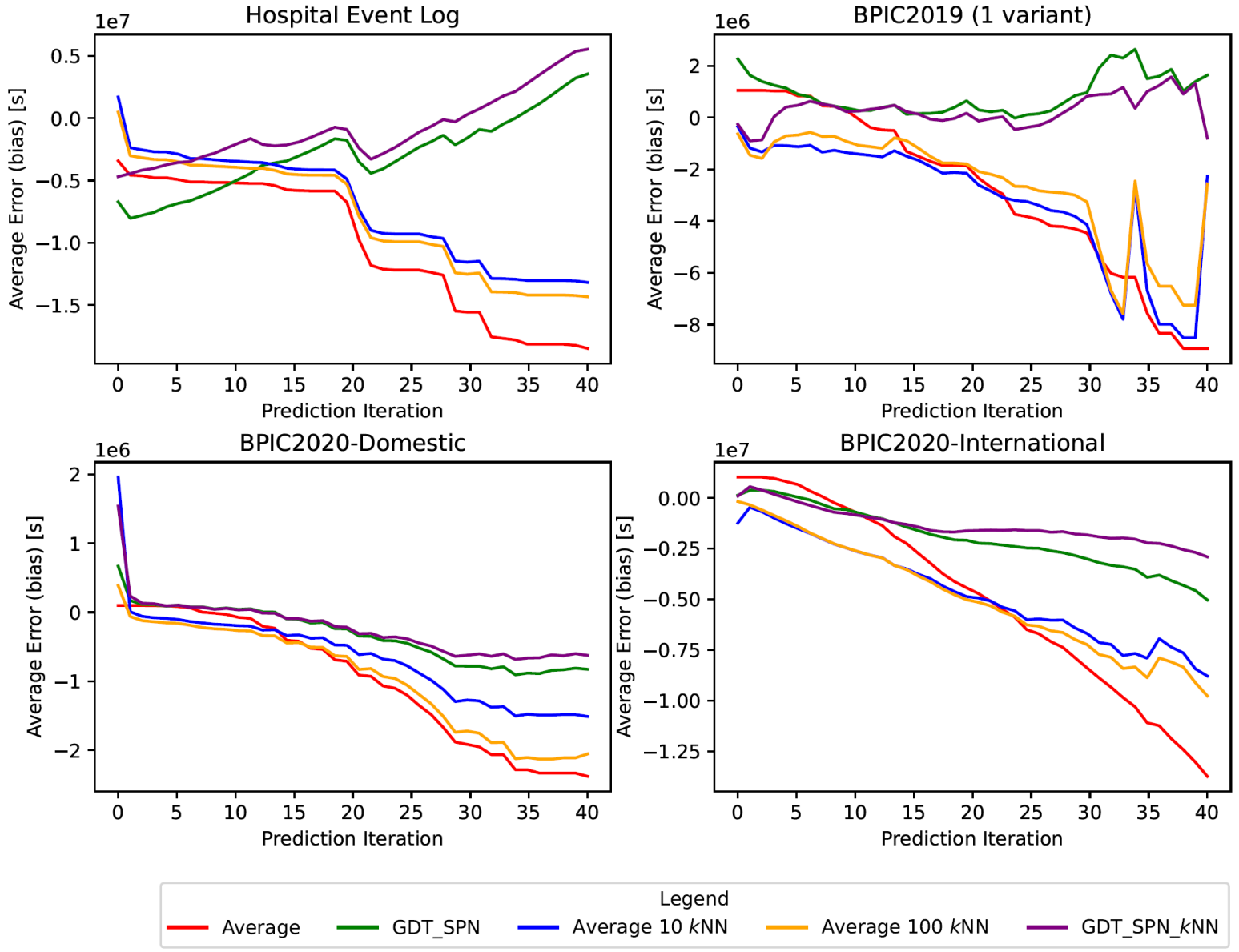
- The average end-time
- A GDT_SPN trained on all available training data
- Average 10 nearest neighbors
- Average 100 nearest neighbors
- **GDT_SPN_kNN** *(with k = 100)*


Hospital Event Log
1e7
RMSE [s]
Prediction Iteration
BPIC2019 (1 variant)
1e6
RMSE [s]
Prediction Iteration
BPIC2020-Domestic
1e6
RMSE [s]
Prediction Iteration
BPIC2020-International
1e7
RMSE [s]
Prediction Iteration
Legend
Average
GDT_SPN
Average 10 kNN
Average 100 kNN
GDT_SPN_kNN

RMSE

- Slight improvement by preselecting 100 nearest neighbors
  - With original GDT_SPN not trailing too much

- For earlier iterations however not much difference between all methods
  - Later in the traces (for a higher iteration), the advantage of incorporating the time already passed on an activity, and possibly other information concerning the activities yet to come, result in more accurate predictions


Hospital Event Log
1e7
Average Error (bias) [s]
Prediction Iteration
BPIC2019 (1 variant)
1e6
Average Error (bias) [s]
Prediction Iteration
BPIC2020-Domestic
1e6
Average Error (bias) [s]
Prediction Iteration
BPIC2020-International
1e7
Average Error (bias) [s]
Prediction Iteration
Legend
Average
GDT_SPN
Average 10 kNN
Average 100 kNN
GDT_SPN_kNN

KU LEUVEN

Conclusion

Conclusion

- We constructed an approach for predicting the remaining time of a business process based on a combination of nearest neighbor selection and the GDT_SPN
- We outperform our four benchmarks, including the original method
- Still explainable *white-box*

Limitations

- For now the distributions are fitted to be normal
  - Not always the case, should be left open
- “Eager” clustering might be smarter in this case
  - Less flexible, but scalable at runtime
- The number of neighbors should be selected more carefully an thoroughly
- The distance metric could be more sophisticated

Future work

- Experiment with different selection approaches, combined with different predictive approaches

- Investigate the influence of *eager* vs *lazy* clustering
- Investigate the influence of choosing *k*
- Try out different distributions for the activity duration

- It used the original discovery technique from Rogge Solit & Weske (2015)
  - Improve the GDT_SPN discovery with more state of the art

Some other future work

- Use a clustering technique in combination with deep learning techniques
  - Theoretically =/= in practice

- Simulate the activity durations of machine learning models
  - See if this provides correct distributions
  - See if truncation (or just providing the already elapsed time) also works