Use of statistical predictors (or machine learning techniques) in the design of real-time systems

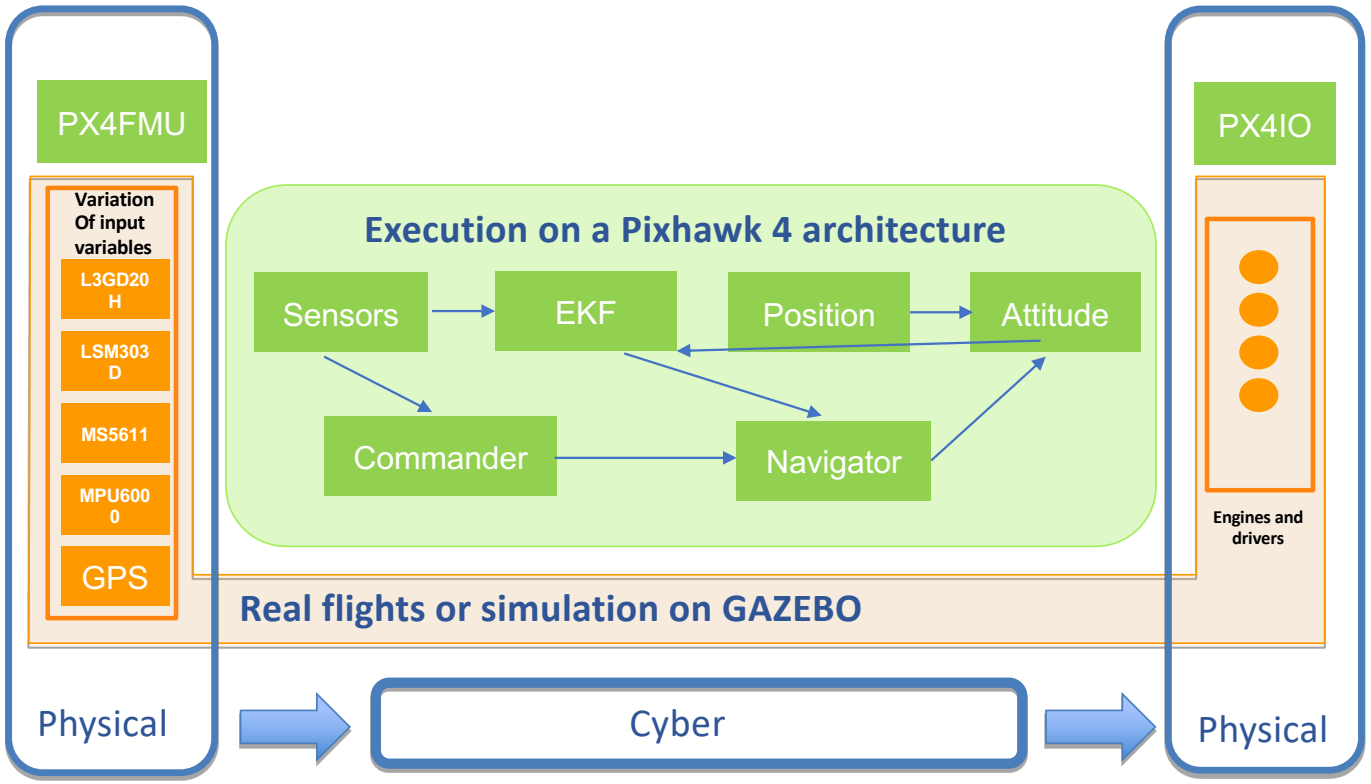
by Liliana Cucu-Grosjean

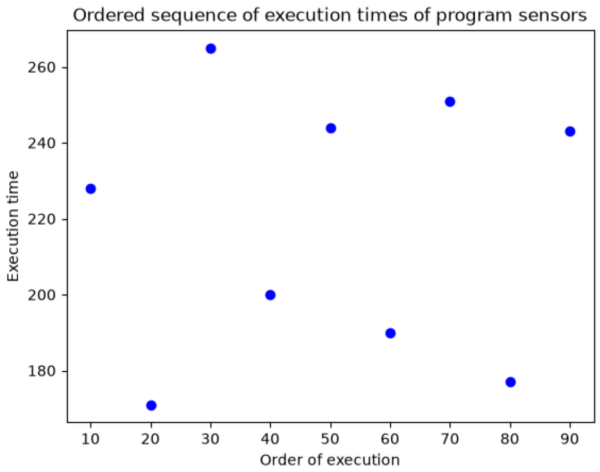
Inria
INVENTEURS DU MONDE NUMÉRIQUE

Outline

- KDBench
- Measurement protocols
- A worst-case statistical estimation
- Time series
- A classification of learning methods
- Relevant Python libraries

KDBench





https://gitlab.inria.fr/kdbench/

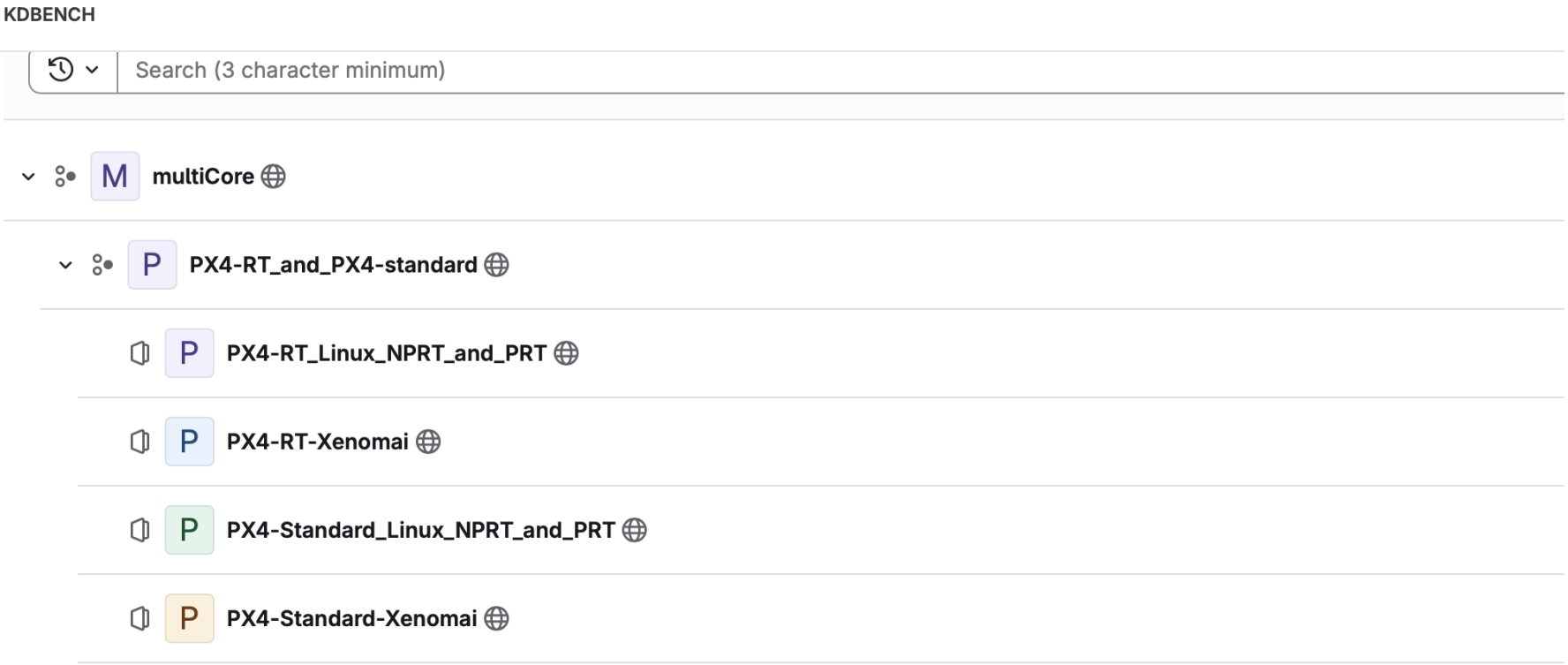
https://team.inria.fr/kopernic/kdbench/

Thanks to

- **KDBench** (PX4-RT programs): based on PX4 open-source programs (ETH Zurich), modified by Mehdi Mezouk, Hadrien Clarke and Masum Bin Alam under the supervision **of Yves Sorel**
- Modifications from event to time triggered during the projects :
  - **Associated Kepler/Giordano team with UFBA (Prof. George Lima's group)**
  - FR FUI20 CEOS led by Thales Communications during the PhD thesis of Slim Ben Amor
  - FR PSPC-2022 STARTREC (led by Easymile)
  - Inria ADT KDBench
  - StatInf/Inria CIFRE PhD theses of Marwan Wehaiba and Ismail Hawila
- On-going energy-oriented evolution: PhD thesis of Myriam Mabrouki
- On-going CPU-GPU evolution: PhD thesis of Hadjer Bendellaa


KDBENCH
Search (3 character minimum)
Name
M multiCore
1 3 1
Created 11 months ago
P PX4-RT_and_PX4-standard
0 4 1
Created 11 months ago
B bouif
0
Created 11 months ago
D data
0
Created 11 months ago
D documents
0
Created 11 months ago
S singleCore
0 3 1
Created 11 months ago
B bouif
0
Created Dec 5, 2024
D Data
0
Created Dec 5, 2024
P PX4-RT
0
Created Dec 5, 2024

KDBench: several OS choices

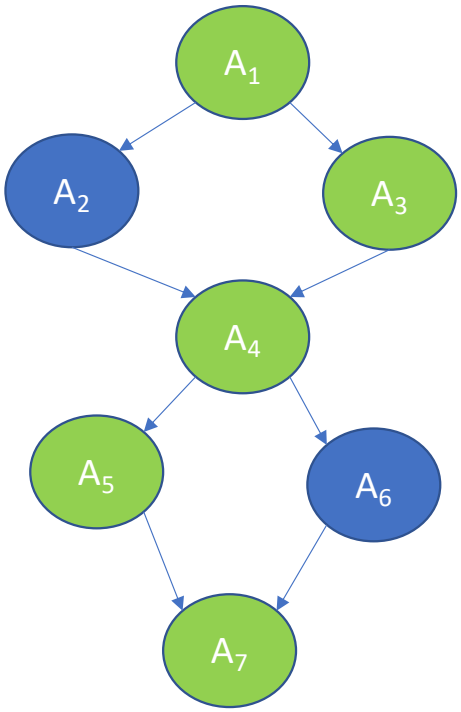


Outline

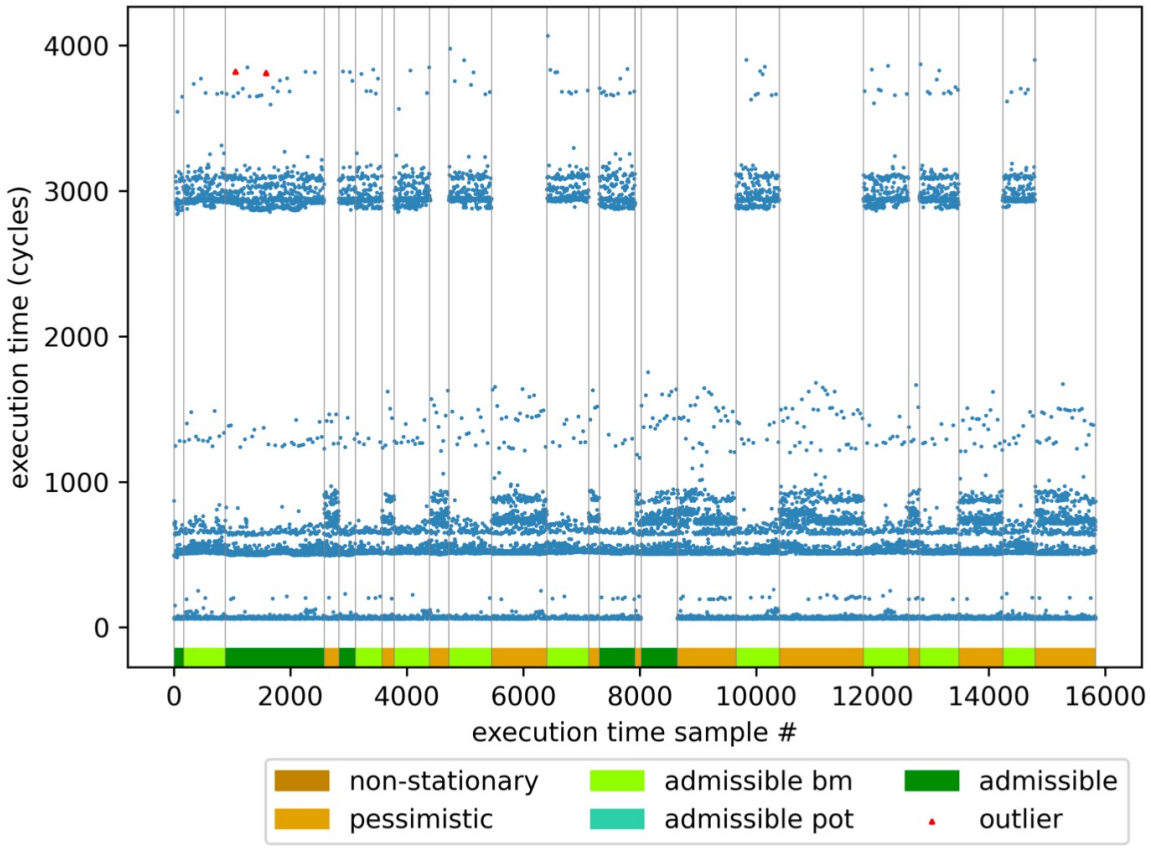
- KDBench
- Measurement protocols
- A worst-case statistical estimation
- Time series
- A classification of learning methods
- Relevant Python libraries

Measurement protocols

- If we are interested in execution times, then we need to understand their sources of variation



EKF

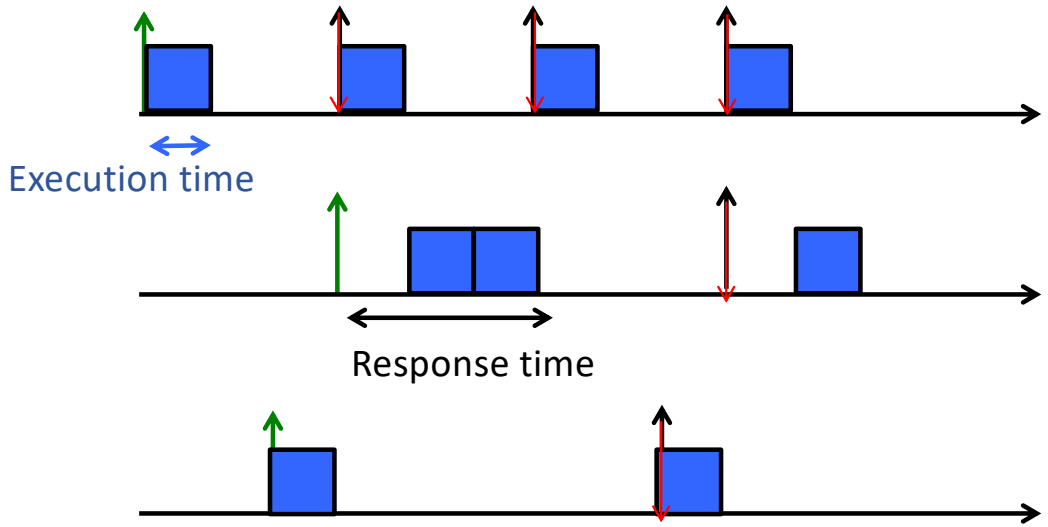


A scheduling only view

Navigator (0, 1, 3, 3)

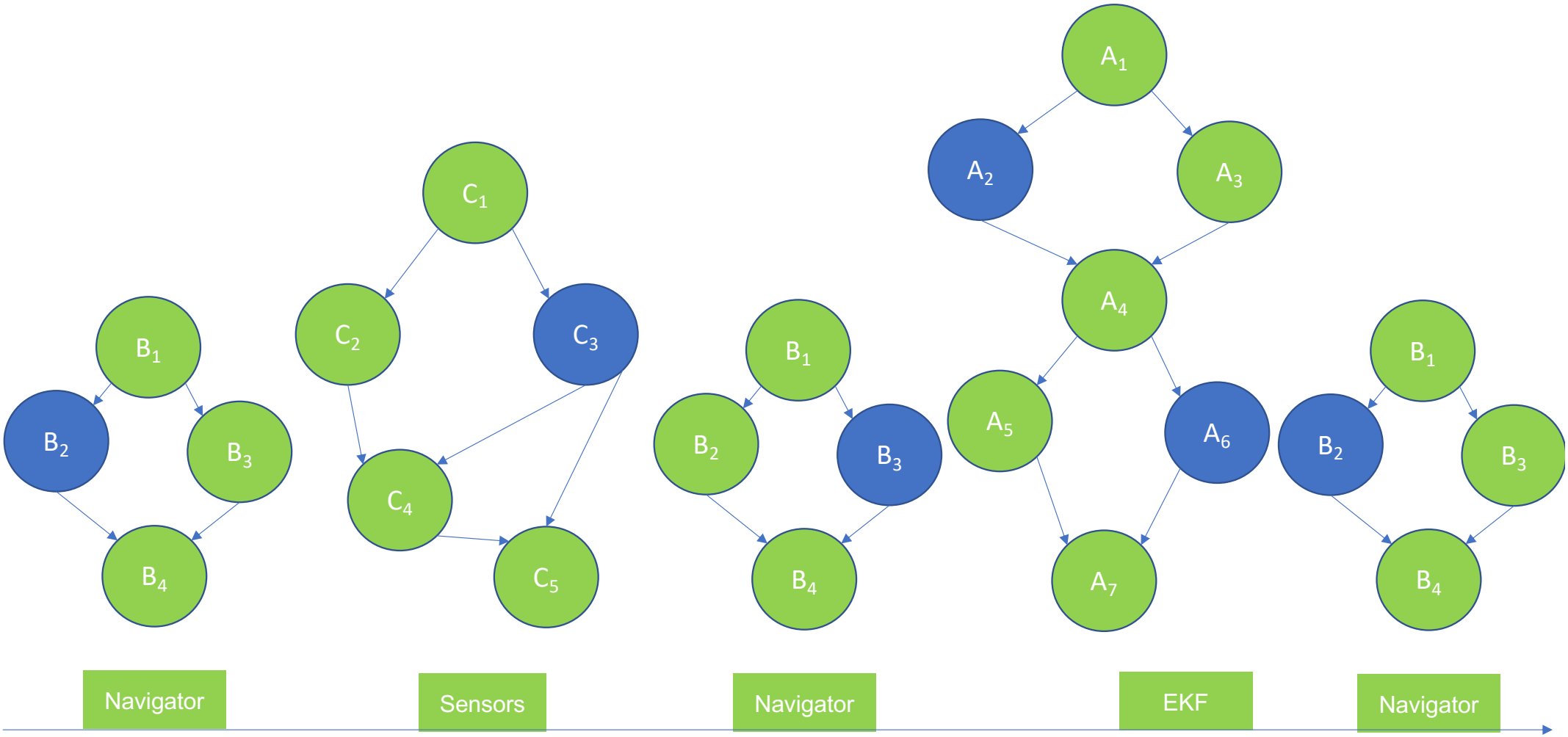
EKF (3, 2, 6, 6)

Sensors (2, 1, 6, 6)



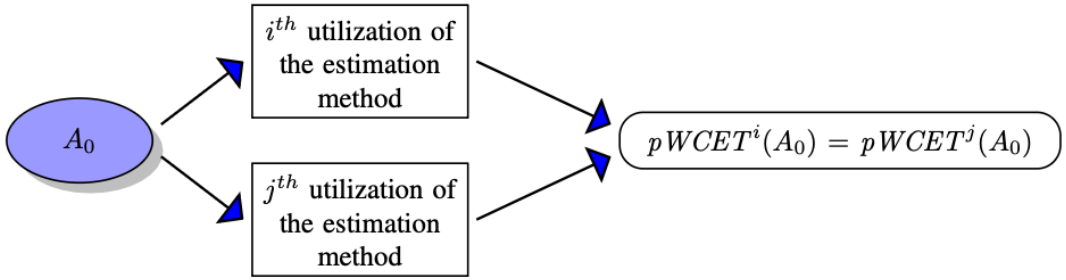
One core, fixed-priority scheduling algorithm

A deeper view within programs

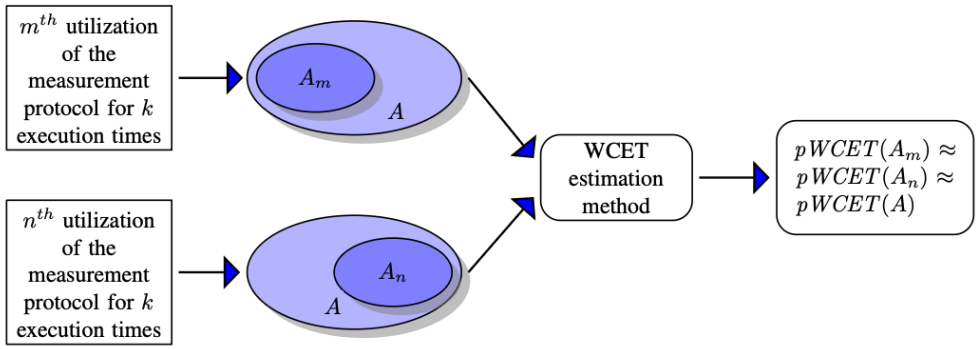


Measurement protocols

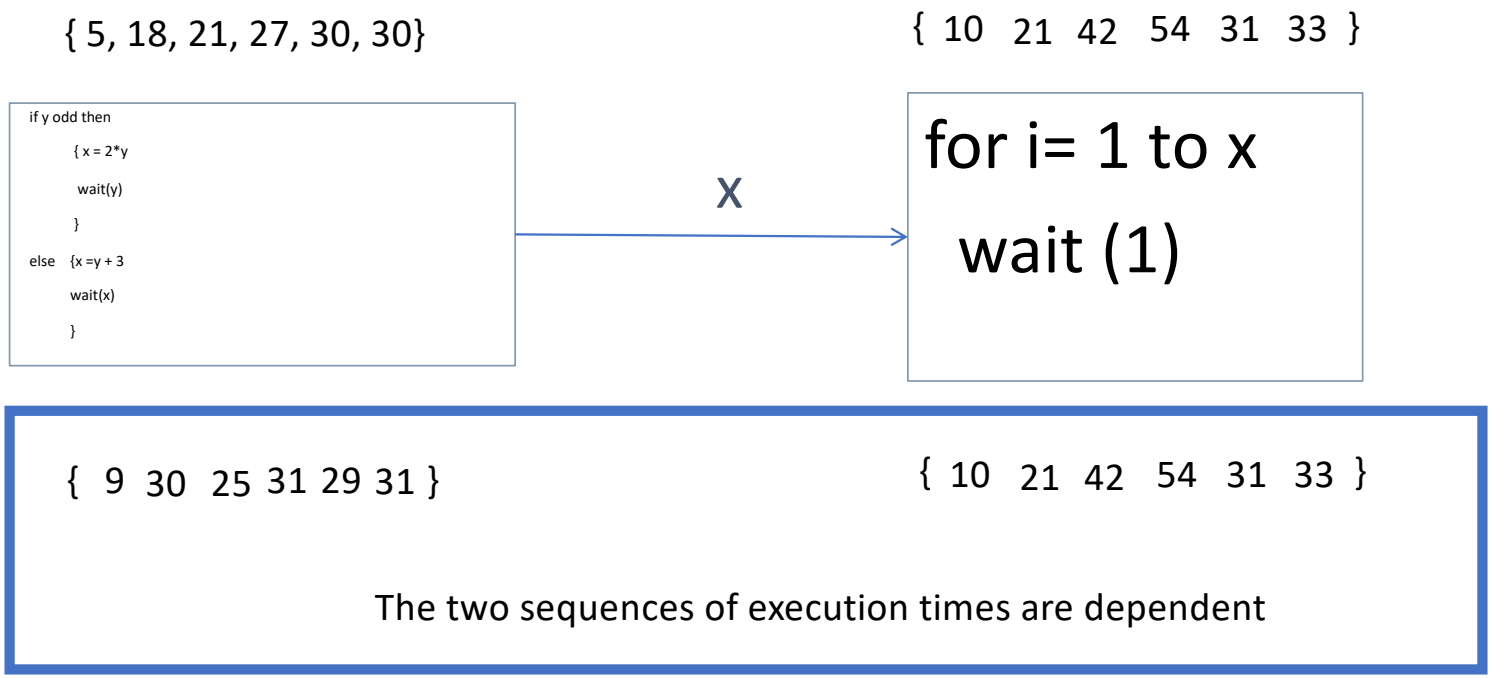
- Reproducibility



- Representativity
- Organisation of experiments



Statistical dependence



Two programs with (functional) dependences

Statistical independence

{ 68, 59, 84, 94, 100, 57}

```
if y odd then
        { x = 2*y
         wait(y)
        }
else    {x =y + 3
        wait(x)
        }
```

{39, 27, 39, 36, 34, 41}

```
for i= 1 to x
  wait (1)
```

{69, 63, 85, 95, 101, 61}

{39, 27, 39, 36, 34, 41}

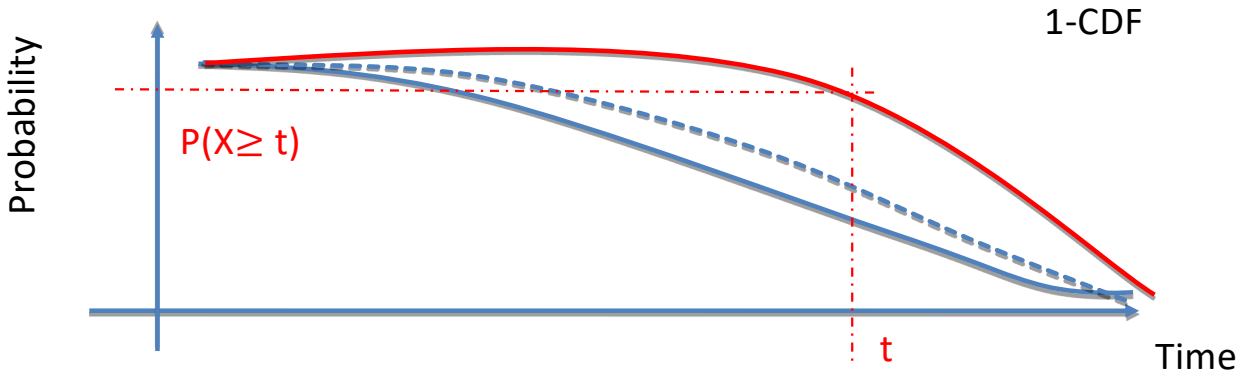
The two sequences of execution times are independent

Two programs that have (functional) dependences

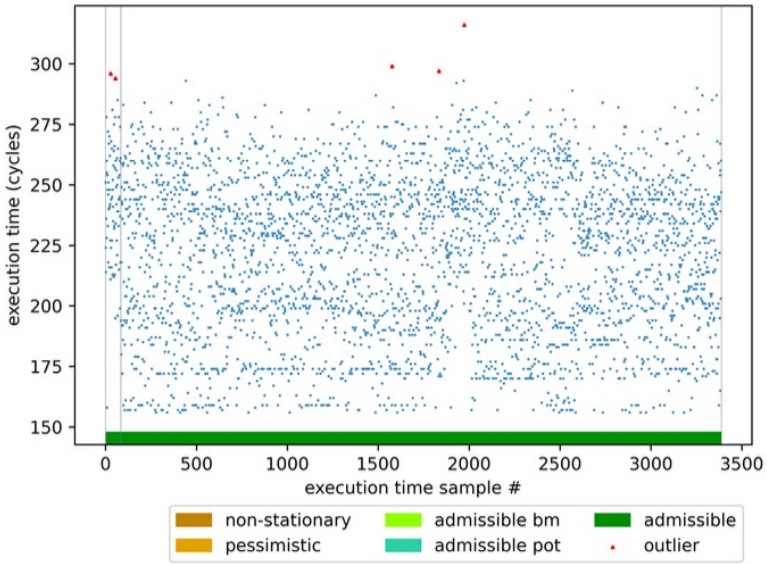
Outline

- KDBench
- Measurement protocols
- A worst-case statistical estimation
- Time series
- A classification of learning methods
- Relevant Python libraries

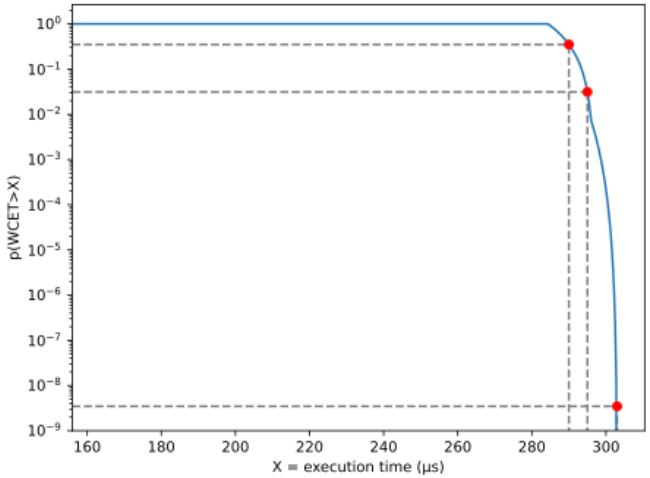
A worst-case statistical estimation



From ordered sequences of values to distributions



(a) Sensors



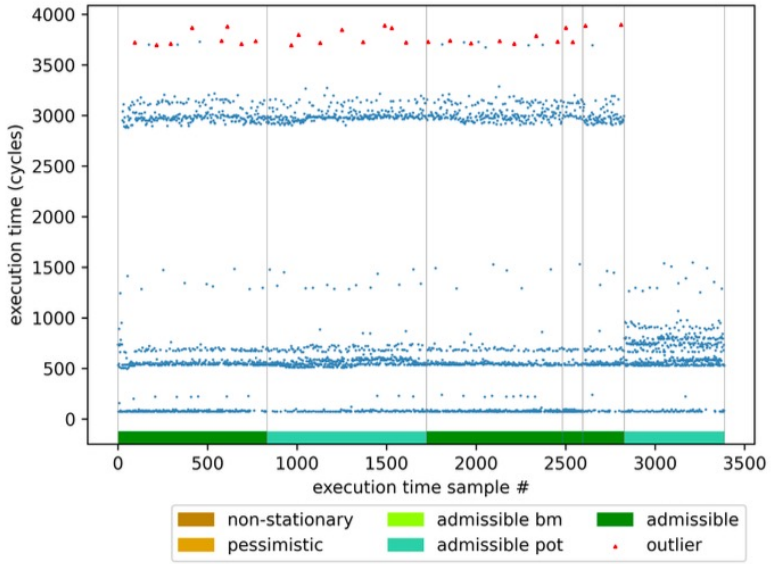
(a) Sensors

Figures from the PhD thesis of Ismail Hawila (https://theses.hal.science/tel-05599445) and Marwan Wehaiba (https://theses.hal.science/tel-04999351)

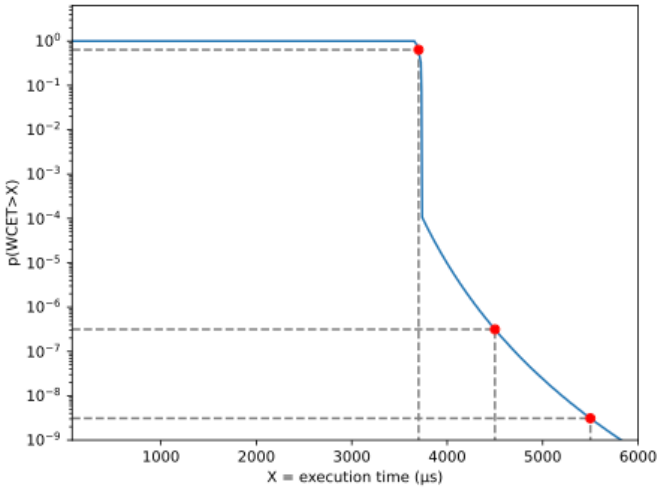
Outline

- KDBench
- Measurement protocols
- A worst-case statistical estimation
- Time series
- A classification of learning methods
- Relevant Python libraries

From ordered sequences of values to distributions (2)



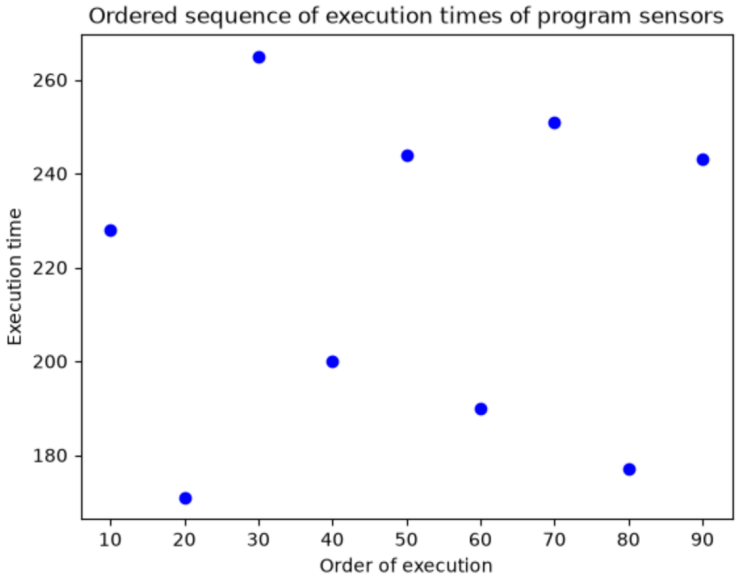
(b) EKF2



(b) EKF2

Time series: the order

- A sequence of data points that occur in a successive order over some period of time




```
import matplotlib.pyplot as plt
execution_times=[228, 171, 265, 200, 244, 190, 251, 177, 243]
order=[10, 20, 30, 40, 50, 60, 70, 80, 90]
fig, ax = plt.subplots()
ax.scatter(order, execution_times, marker="o", color="blue")
ax.set_xlabel("Order of execution")
ax.set_ylabel("Execution time")
ax.set_title("Ordered sequence of execution times of program sensors")
plt.show()
```

https://gitlab.inria.fr/kdbench/singlecore/data/-/blob/main/short_mission/scenario1/Execution_times.csv?ref_type=heads

Time series: some definitions

- Objective: identification of anomalies and/or outliers
- Anomaly is a deviation from the general distribution of the data, a single observation (a point) or a subsequence that deviates greatly from that distribution and that forms only a small minority of the series. Anomalies usually signal an error or a noteworthy change such as a malfunction, a fraud or an unusual spike
- Outlier is an extreme value outside the general distribution that is not necessarily tied to a failure and may be noise, a measurement error or a rare value
- A slicing window is a sub-sequence (x_{t+1}, … , x_{t+L}) of consecutive observations that a model receives as input

A (partial) classification of learning methods*

- Forecasting
- Detection and anomaly prediction
- Statistical and Threshold Based Methods
  - ✓ Z-score and statistical thresholding
  - ✓ Extreme value theory
  - ✓ Autoregressive forecasting models: ARIMA and VAR
  - ✓ Markov models: hidden and extensible]
- Supervised Machine Learning
  - ✓ Linear and probabilistic baselines: logistic regression and naive Bayes
  - ✓ Tree ensembles: decision trees, random forest, XGBoost and AdaBoost
  - ✓ Support vector machines and one-class SVM
  - ✓ k-nearest neighbors
- Deep Learning: Forecasting Based Detection
  - Convolutional networks: CNN, TCN and ResNe
  - Graph attention networks

* **Classification proposed by Kyrylo Hrynevych during his MSc internship this summer**

A deeper look

Anomaly Detection and Forecasting Python Libraries

Kyrylo Hrynevych

July 2026

Library	Approach	Main drawback	Potential improvement
Anomaly detection			
River	Online anomaly scoring with `GaussianScorer`. The stream is processed sequentially and WCEC candidates are selected from high-energy values using anomaly scores.	The score is local and distribution-based. So even after filtering for high values, it may rank points by statistical surprise instead of actual worst-case energy relevance, limiting precision when many high but normal-appearing values exist.	To implement a custom one-sided high-tail score or combine River scores with EVT thresholds. Add more context.
Alibi Detect	Offline outlier scoring with Isolation Forest. The model assigns an isolation score and WCEC candidates are selected among high-energy points.	Isolation Forest finds points that are easy to isolate and not necessarily meaningful high-energy spikes. With mostly one-dimensional energy data, it can behave like an ordinary ranking method and might not distinguish real WCEC events from dense high-value regions.	To use vectors with more features: energy, CPU/memory frequency, rolling mean, rolling maximum, rolling standard deviation and local spike density. And to adjust final candidates with EVT or a high-tail threshold.
Anomaly forecasting			
Darts	Forecasts future energy values from the training part, then selects predicted WCEC candidates from high forecasted values and fits EVT on them.	It tends to predict the average trajectory, it does not naturally forecast rare spike timing, so predicted WCEC values may be flat, too smooth or poorly aligned with real high-energy events.	To use probabilistic or quantile models instead of point forecasts. And to include time-varying covariates and use rolling forecasts with shorter horizons. To model spike probability separately from the expected energy value.
PyTorch Forecasting	Trains a Temporal Fusion Transformer (TFT) with energy and covariates, then uses upper quantile forecasts to select predicted WCEC candidates.	Even with TFT, spike timing remains weak if covariates are constant or not causally related to spikes. The model may estimate a high-risk level but still produce nearly constant upper quantiles and poor temporal precision.	To add informative execution-level covariates such as phase, hardware counters, recent volatility, lagged values and local maxima. To use strict rolling prediction and train directly on WCEC probability or high-quantile loss.

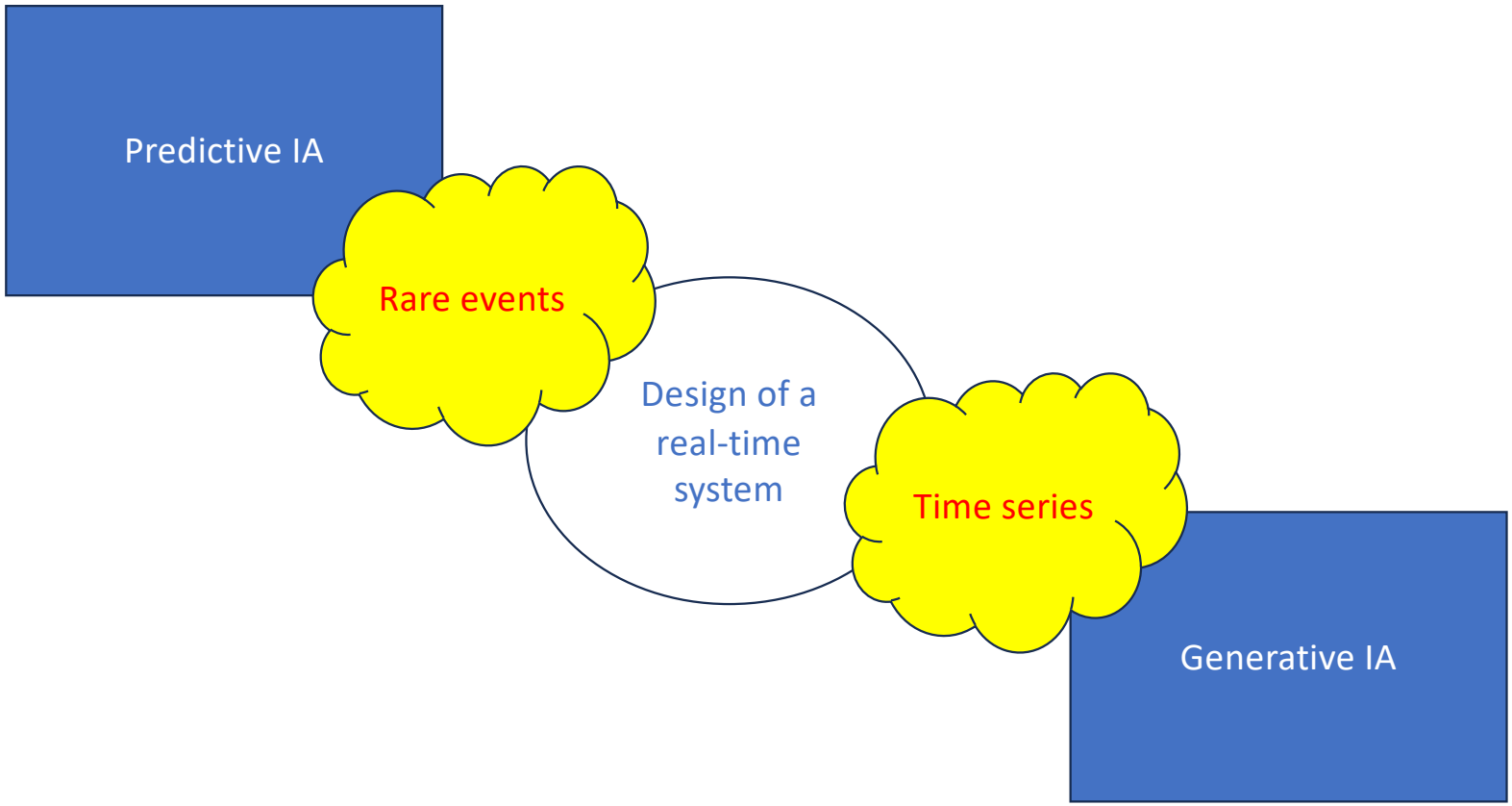
Relevant Python libraries

- Based on the analysis of Kyrylo Hrynevych on data proposed by Myryam Mabrouki
  - ✓Anomaly detection: River, Alibi Detect
  - ✓Anomaly forescating: Darts, Pytorch Forescasting
- Systematically results compare against Extreme Value Theory estimations
  - ✓https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.genextreme.html
  - ✓from scipy.stats import genextreme as gev

- Some sanity checks to underline when dealing with time series

https://who.rocq.inria.fr/Liliana.Cucu/ETR-2026.py

Open problems



Thank you for your attention



liliana.cucu@inria.fr