Deriving Timing Properties from System Traces Using Data-driven Techniques

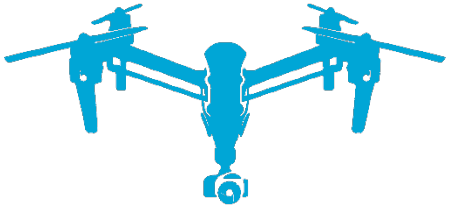
Șerban Vădineanu

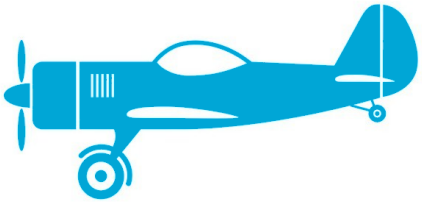
Supervisor: Mitra Nasri

TU Delft

What systems?

- **Real-Time Systems (RTS)**:
  - the system's correctness depends on both its **functional** and **temporal** correctness.





Why deriving properties?

Embedded systems' complexity is growing

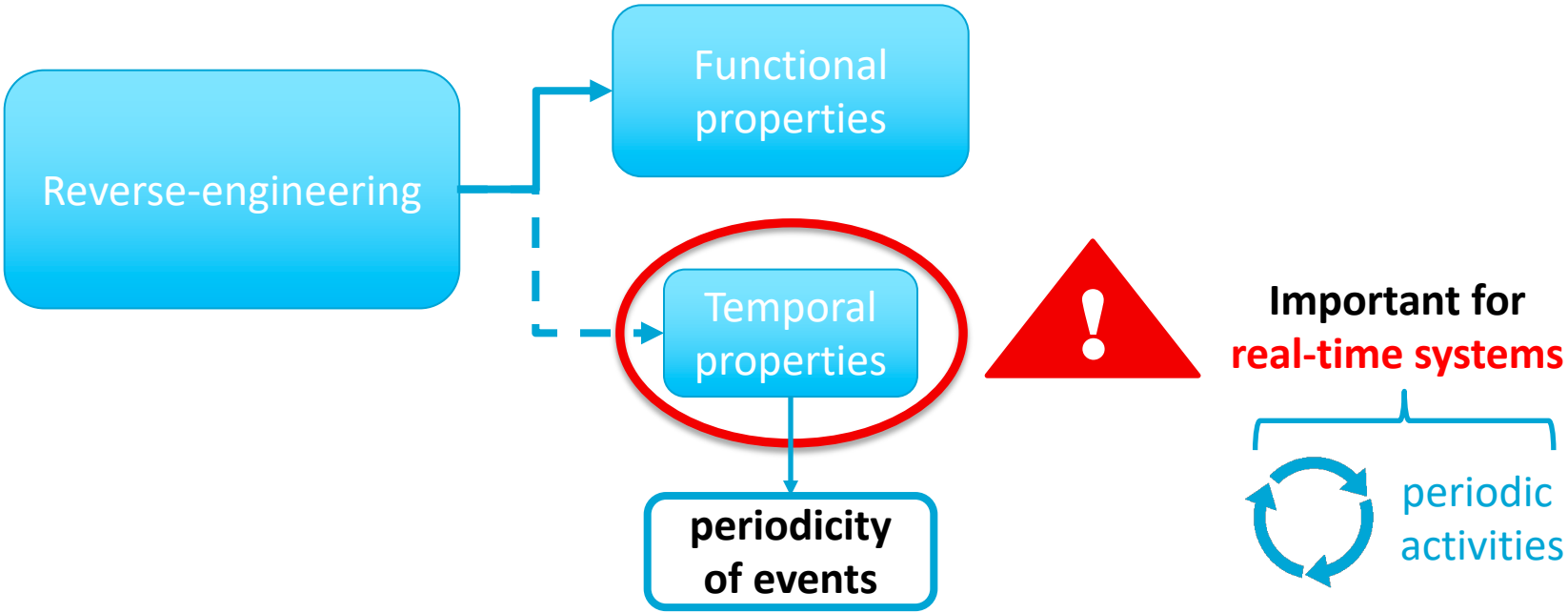
Complexity results in more bugs

Complexity poses challenges on security

There is a need for tools to understand the system's **behavior**

Reverse-engineering

Which properties?



Why is period inference challenging?

★☆☆

No operating system
Only one task

★★☆

Lower-priority task
in a real-time operating system

★★★

High-priority aperiodic tasks,
sporadic tasks,
missed jobs,
release jitter

Our work

task– a functionality of the system, which periodically releases instances (**jobs**)

jitter– maximum deviation of the release time among all jobs of a task

TU Delft

What has been done?

Period estimation

- periodogram
- autocorrelation

} Fourier transform

Machine learning for reverse-engineering real-time systems

- neural networks
- k-nearest neighbors

Reverse-engineering timing properties of real-time systems

- only two works on period inference:
  - **PeTaMi**
  - Fourier transform-based

NO machine learning solution for period inference

[1]O. Iegorov, R. Torres, and S. Fischmeister. “Periodic task mining in embedded system traces”. *Real-Time and Embedded Technology and Applications Symposium*, pp. 331-340, 2017.

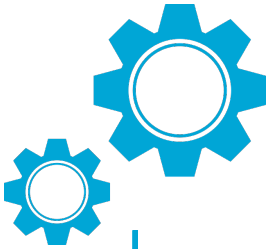
[2]C. Young, H. Olufowobi, G. Bloom, and J. Zambreno. “Automotive intrusion detection based on constant CAN message frequencies across vehicle driving modes.” *ACM Workshop on Automotive Cybersecurity*, pp. 9-14, 2019.

What is our aim?

Trace

Machine-learning based technique

Regression-based



- high accuracy
- low memory consumption
- fast
- robust

Period (continuous value)

1. Can we use regression-based solutions for the period inference problem?

robustness– the solution works even when the assumptions do not hold

PeTaMi

- Period mining from real-time systems traces.
- Mines intervals between events and checks if they become periodic.
- Not learning based.

Pros ✓

1. **good performance** for harmonic periods
2. does **not require** additional learning

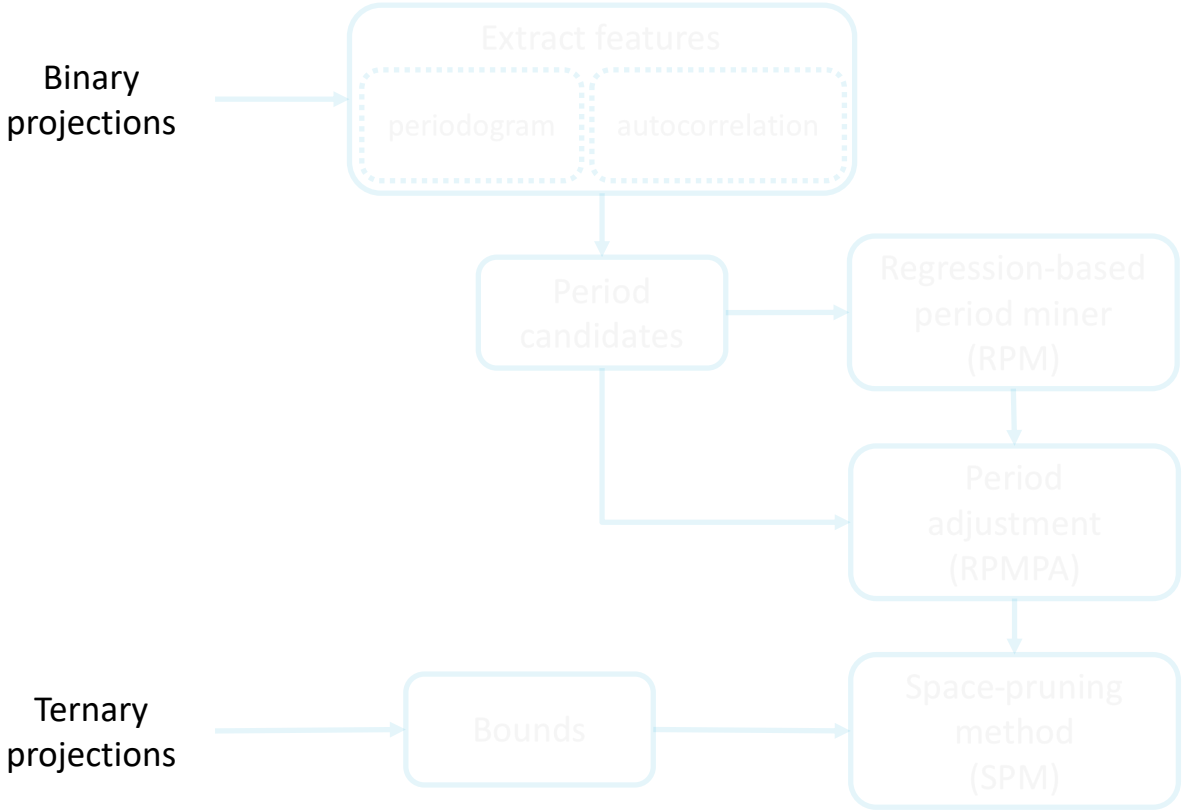
Cons ✗

1. **poor performance** for non-harmonic periods
2. **not robust** if there is execution time variation or release jitter
3. **slow** for longer traces

harmonic periods– every period divides all other smaller periods

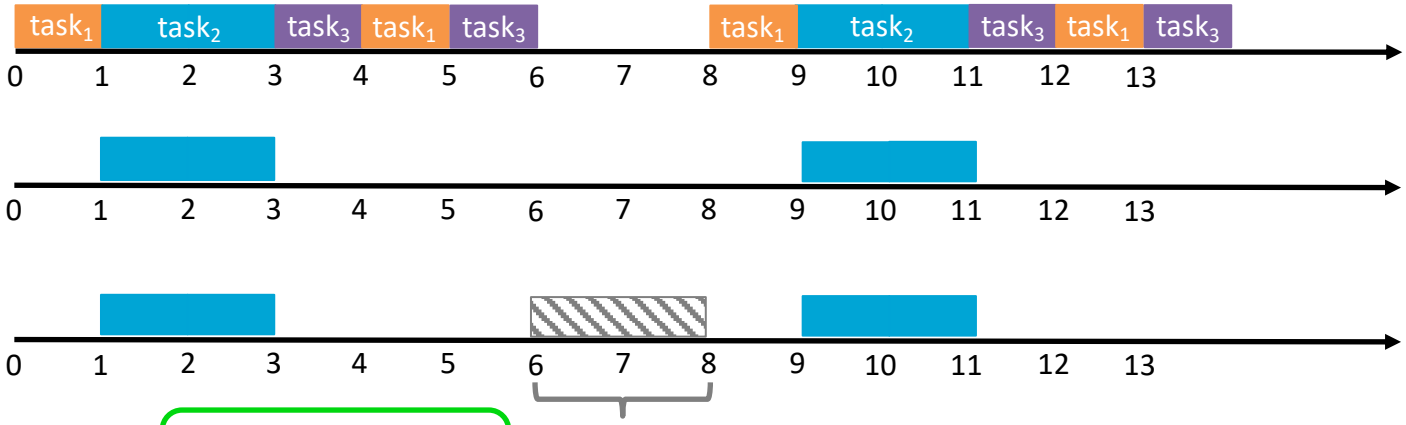
O. Iegorov, R. Torres, and S. Fischmeister. “Periodic task mining in embedded system traces”. *Real-Time and Embedded Technology and Applications Symposium*, pp. 331-340, 2017.

Work in a nutshell



What is a projection?

Schedule



Binary projection of $task_2$

Ternary projection of $task_2$

Input

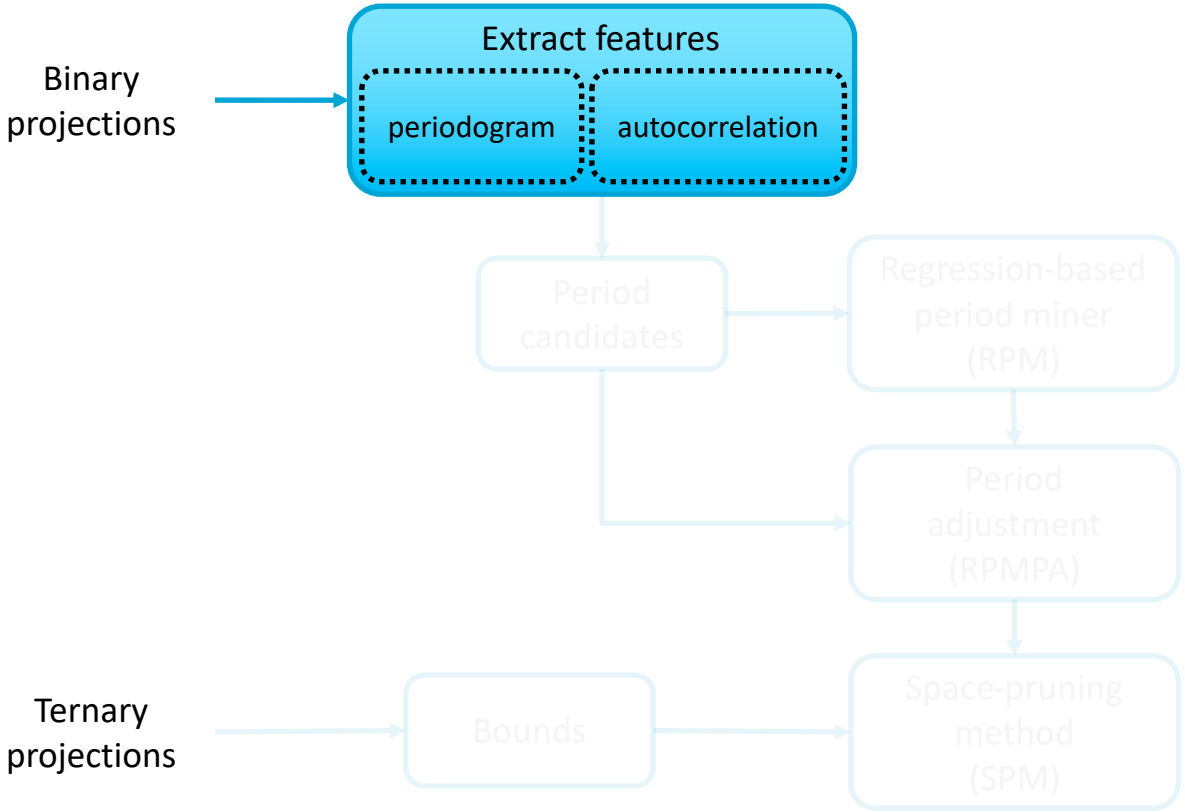
No information about

- other tasks
- scheduling policy

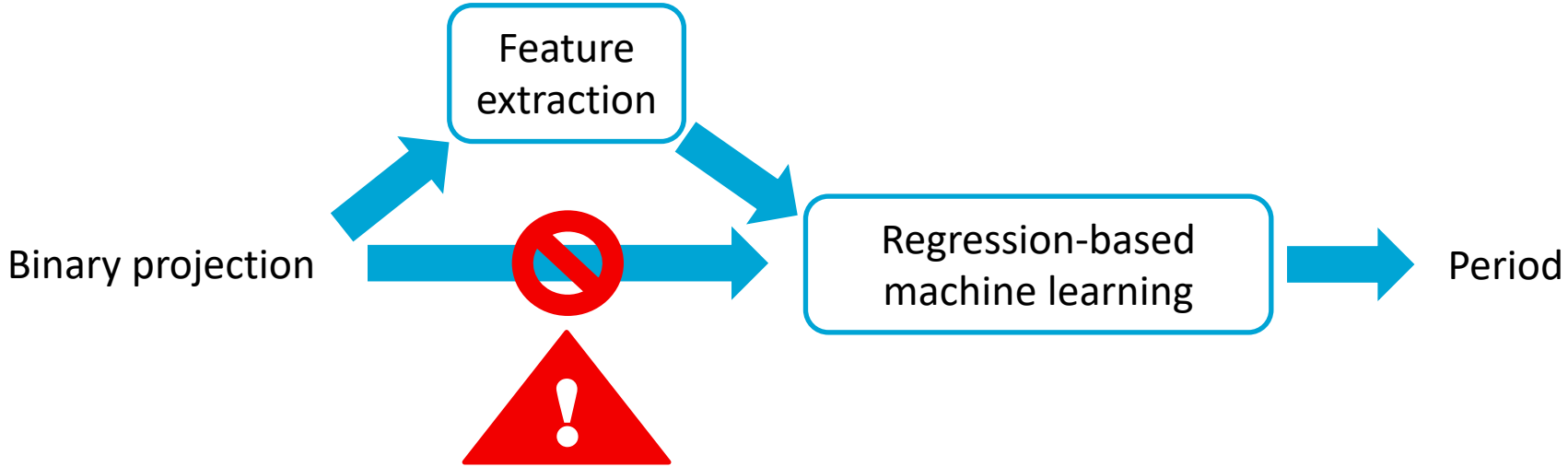
schedule– a particular assignment of tasks to the processor(s) and time intervals. It determines the task execution sequence.

scheduling policy– the policy by which jobs are scheduled

Work in a nutshell

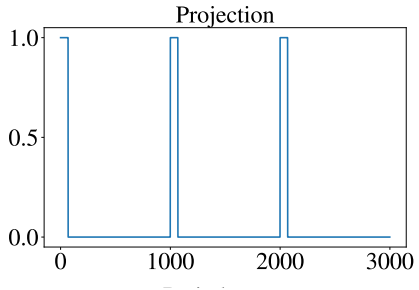


Why extracting features?



- very high dimensionality
- variable-length input

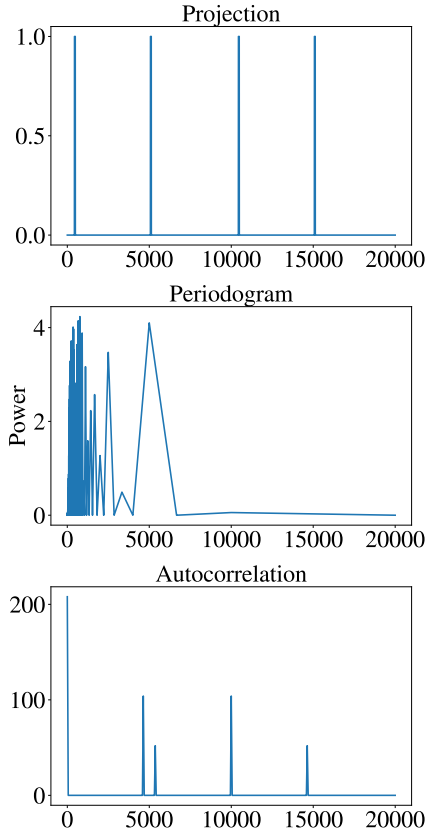
Feature extraction



- an estimate of the spectral density of the signal
- provided by the squared length of each Fourier coefficient of the signal
- examines how similar a sequence is to its previous values for different time lags
- inverse Fourier transform of dot product between the signal and its conjugation

ignore the first peak

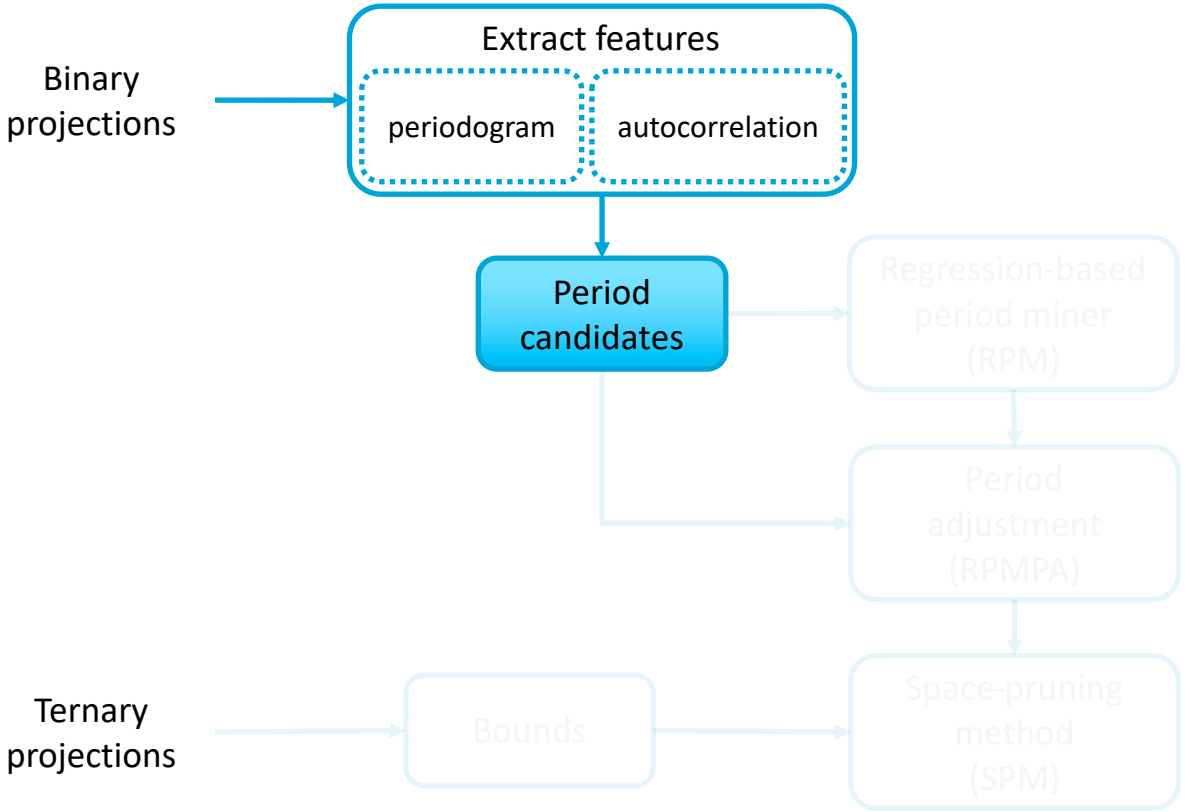
Challenges of feature extraction



- Have multiple peaks
- The highest peak is not necessarily the answer

We decided to extract **period candidates** and employ regression models

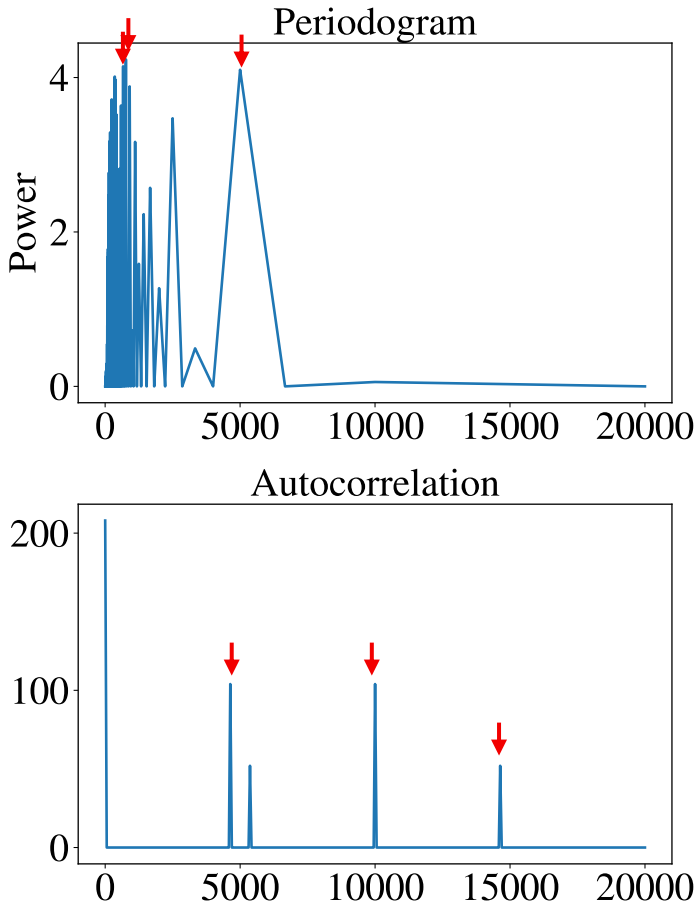
Work in a nutshell



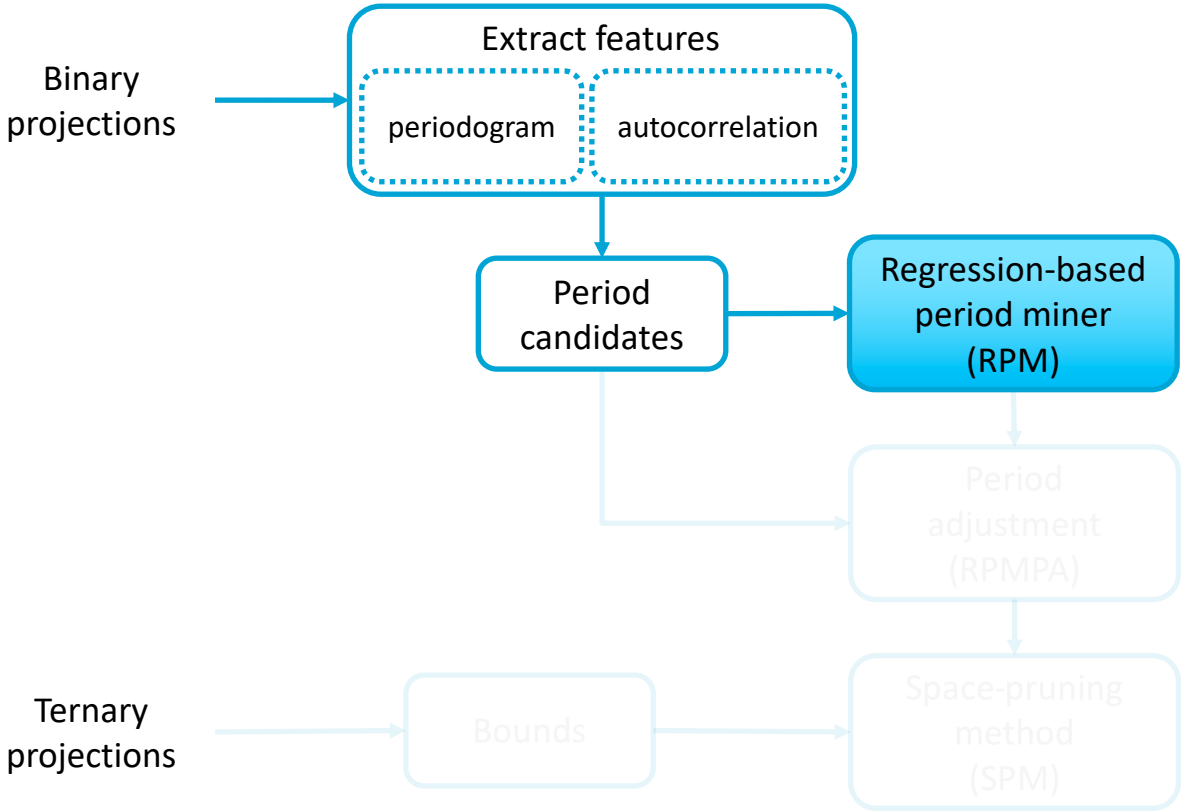
Period candidates

- top 20 periods from the **periodogram**
- top 20 periods from the **autocorrelation** as **candidates**

- top 3 periods from both as **features** for regression



Work in a nutshell



Regression-based period miner (RPM)

Top 3 periods from periodogram
Top 3 periods from autocorrelation

→ Regression-based machine learning → Period

2. Which regression algorithm would result in a better accuracy for inferring periods?

M. Fernández-Delgado, M. S. Sirsat, E. Cernadas, S. Alawadi, S. Barro, and M. Febrero-Bande. "An extensive experimental survey of regression methods". Neural Networks, pp. 11-34, 2019

Which regression algorithms?

cubist

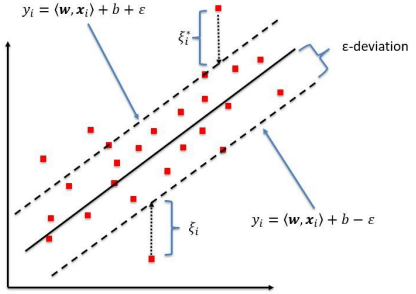
extremely randomized regression trees (extraTrees)

gradient boosting (gbm)

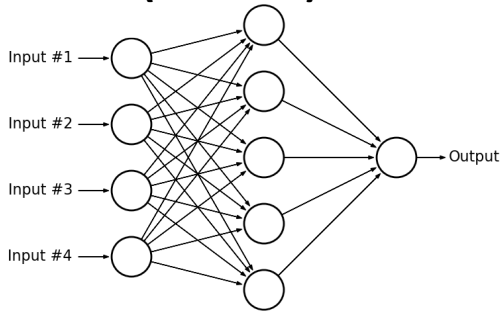
Bayesian additive regression trees (bartMachine)

Tree-based algorithms

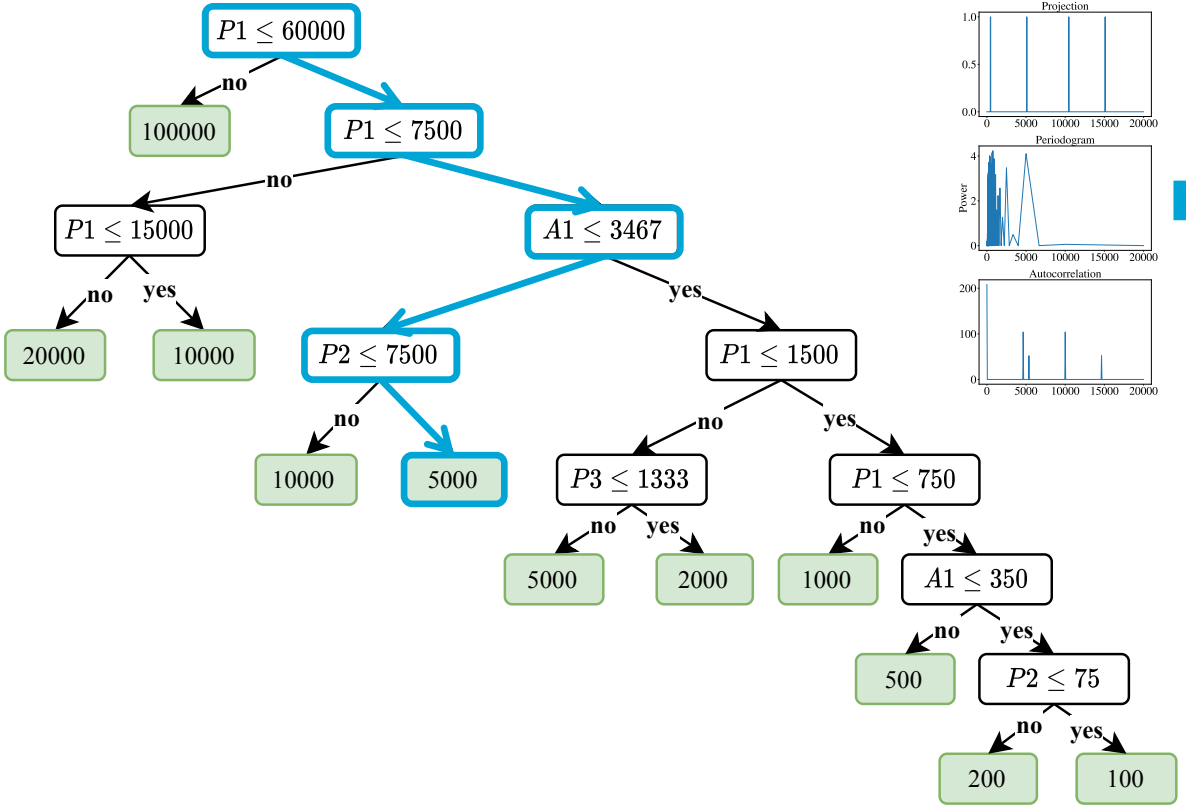
support vector regression (svr)



averaged neural networks (avNNet)



What is a regression tree?



period=5000

P1=769, P2=666, P3=5000,
A1=4635, A2=10000, A3=5365

P1, P2, P3 – top 3 periods from periodogram
A1, A2, A3 – top 3 periods from autocorrelation

Which regression algorithms?

cubist

collapses the tree into a set of rules

extremely randomized regression trees

trains multiple trees averages the output

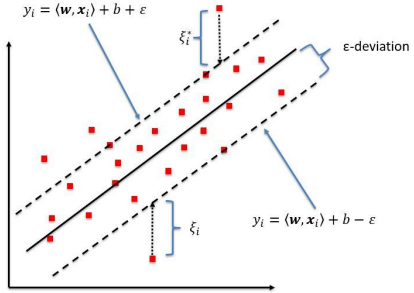
gradient boosting

creates a tree to minimize the error of previous trees

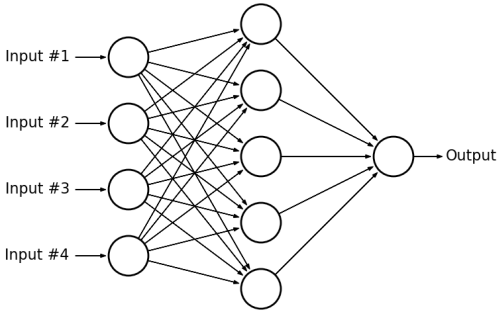
Bayesian additive regression trees

similar to gbm has prior probabilities for tree size + leaf values

support vector machine

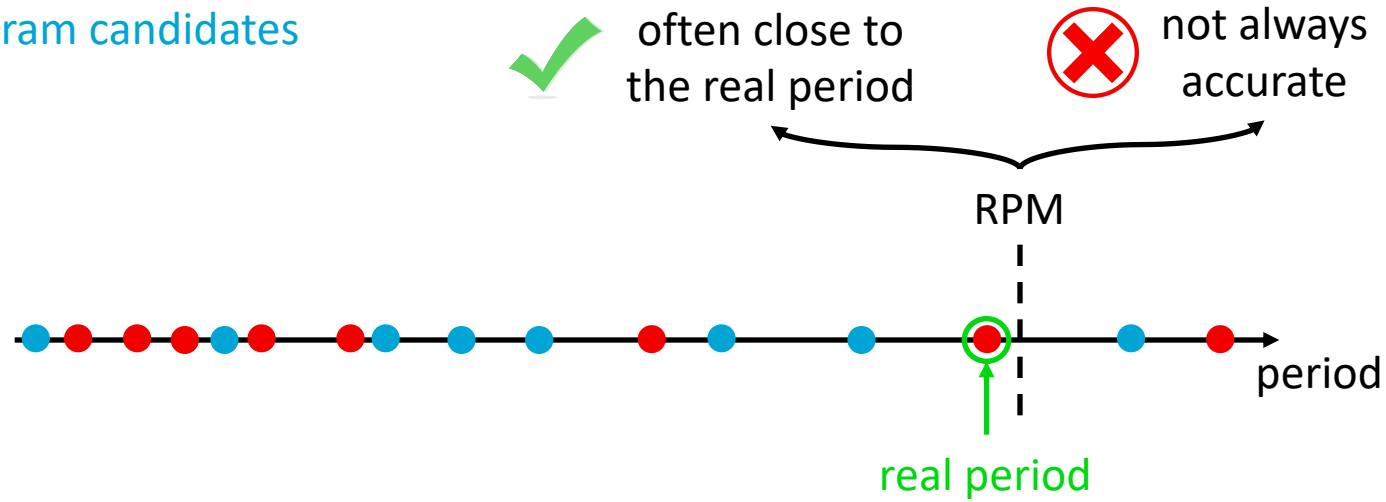


averaged neural networks



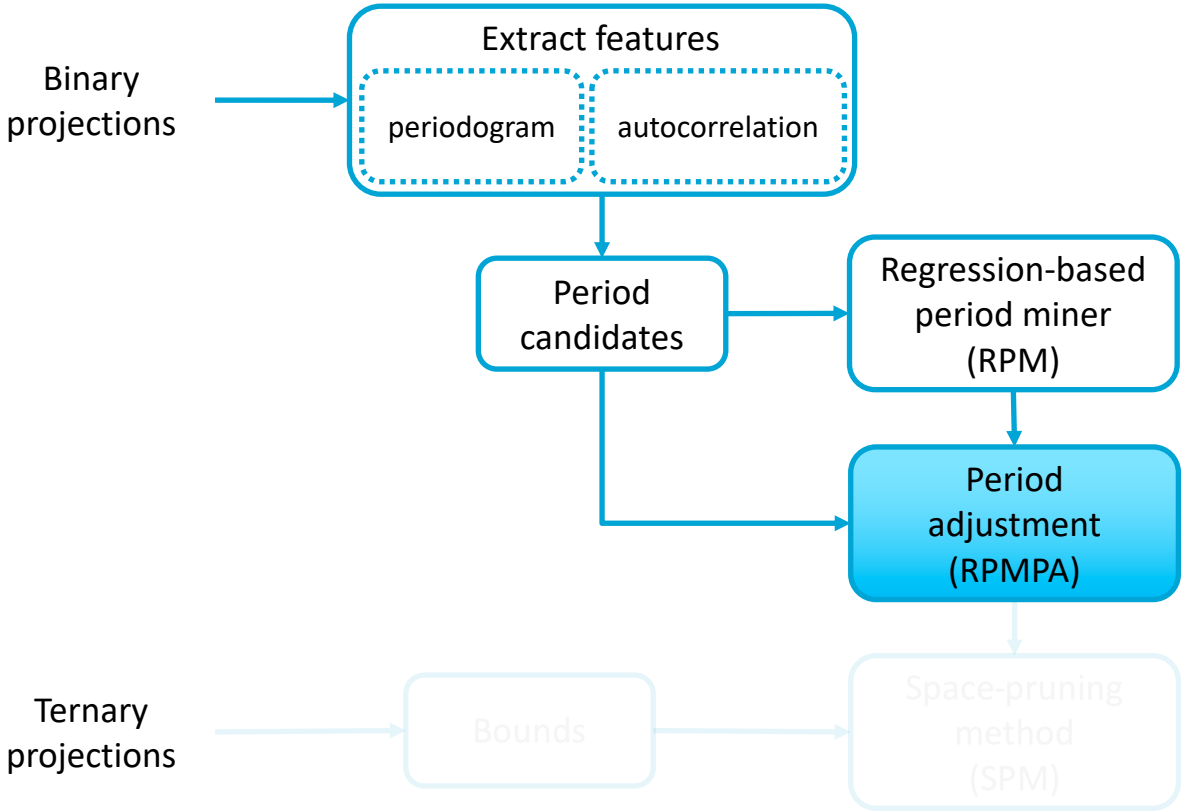
Observations

- autocorrelation candidates
- periodogram candidates



RPM – regression-based period miner

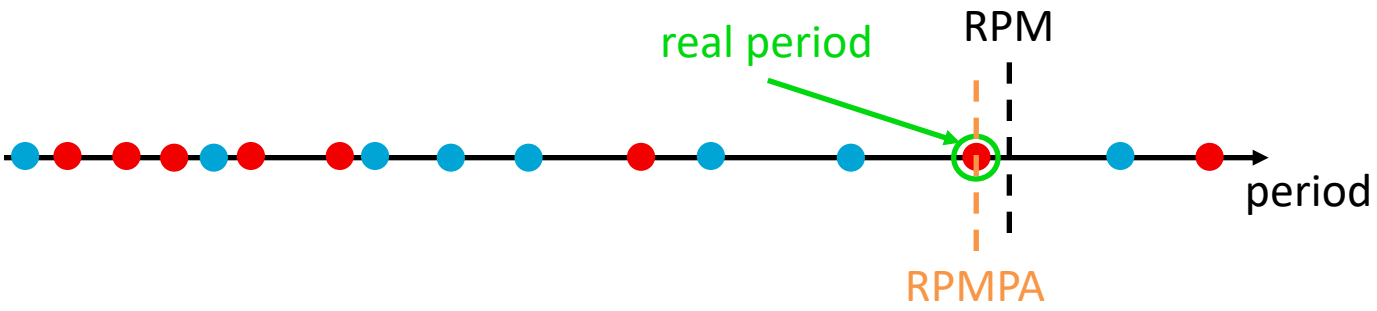
Work in a nutshell



Period adjustment (RPMPA)

1. Create candidate list from top 20 values of **periodogram** **&** **autocorrelation**

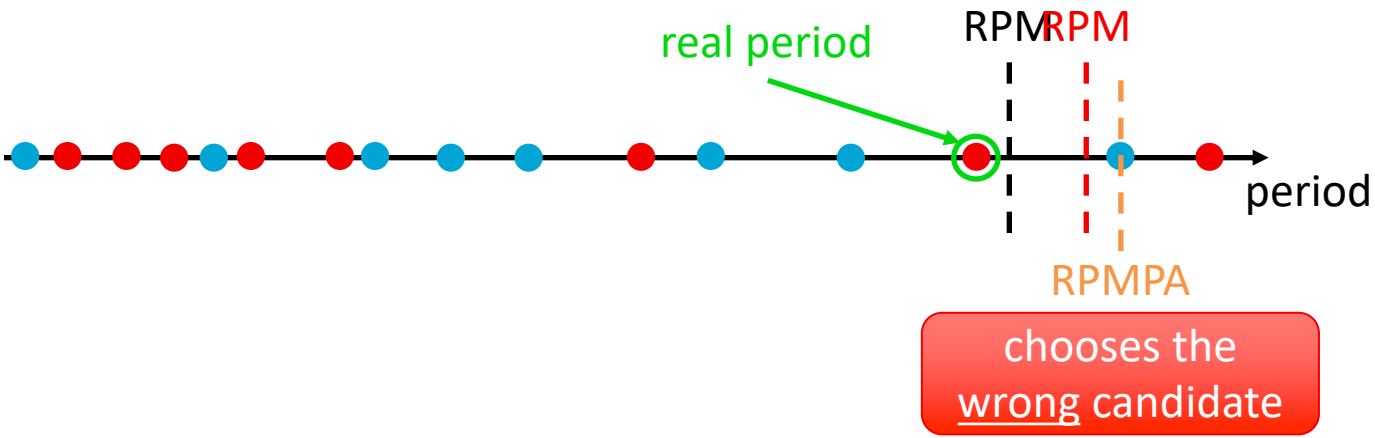
2. Choose the candidate closest to what RPM estimates



RPM – regression-based period miner
RPMPA – regression-based period miner with period adjustment

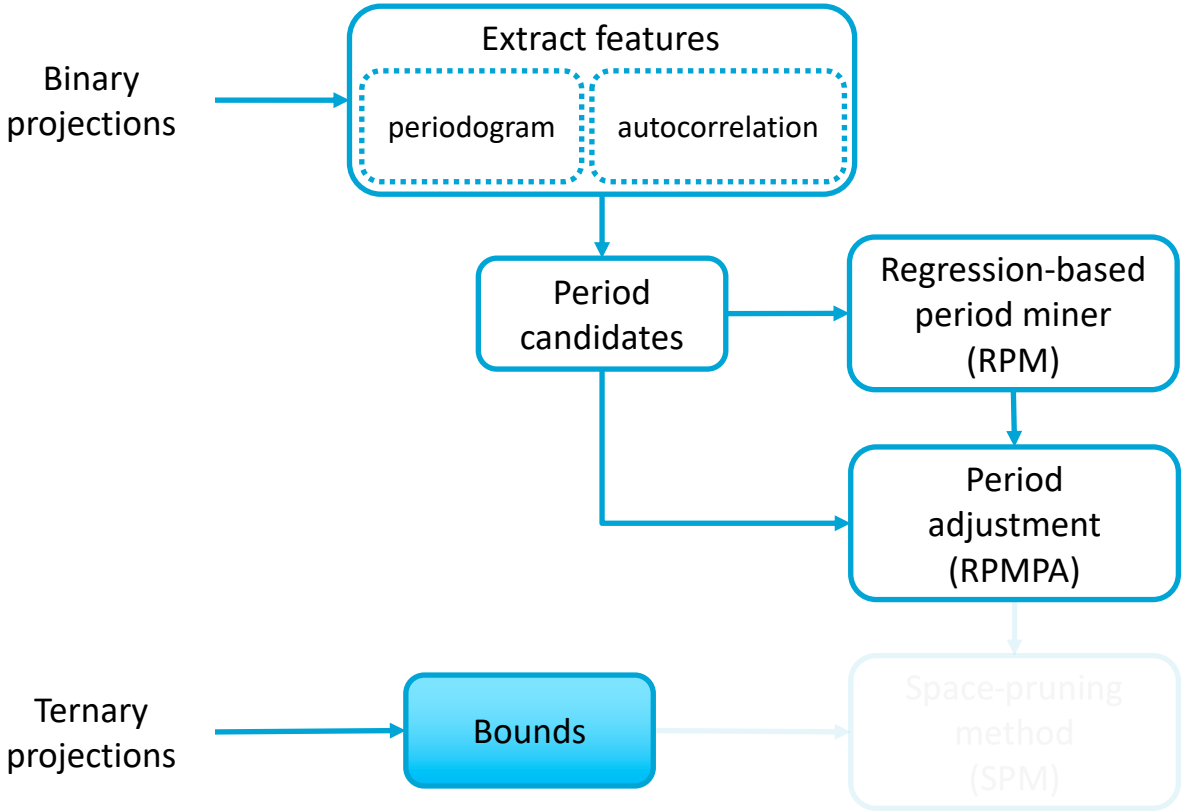
What can go wrong?

3. Can we derive a pair of bounds to restrict candidate values?



RPM – regression-based period miner
RPMPA – regression-based period miner with period adjustment

Work in a nutshell

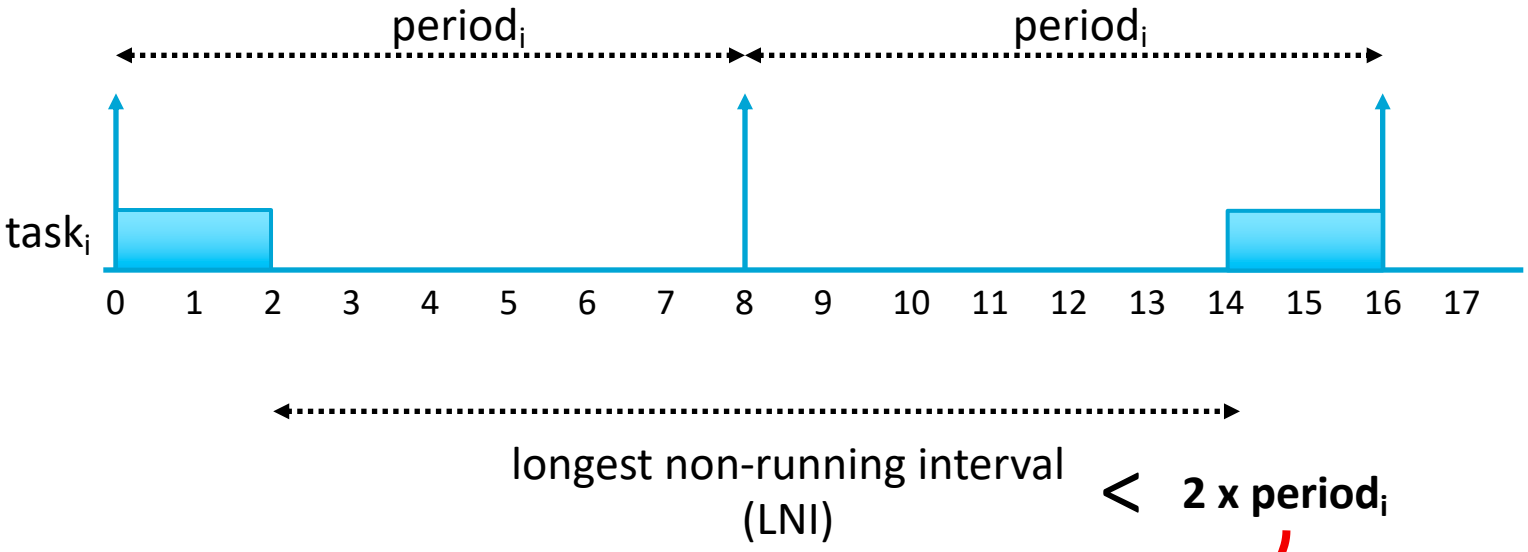


Bounds on the period

1. **Lower bound** – based on longest non-running interval from a projection.

2. **Upper bound** – based on knowledge about the *idle time.*

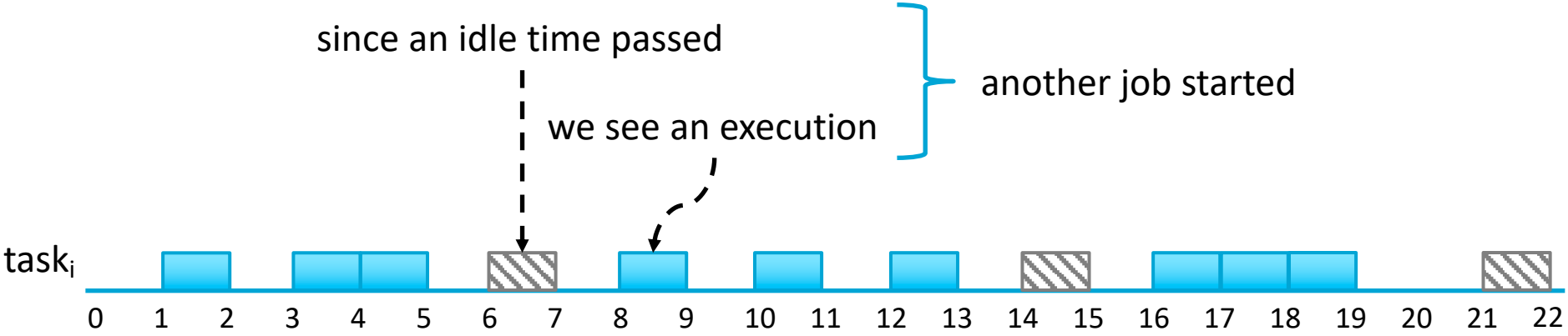
Lower bound derivation



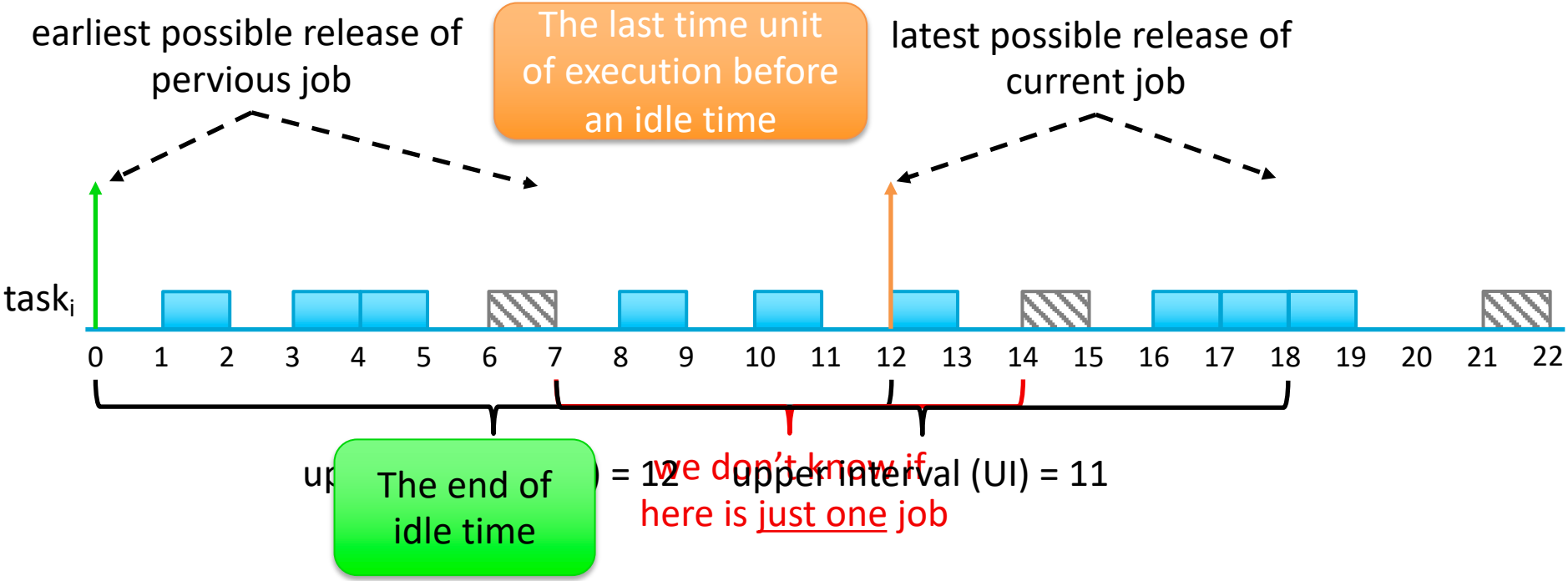
period$_i$ = 8
execution$_i$ = 2
deadline$_i$ = period$_i$

$$period_i > \frac{LNI}{2}$$

Upper bound derivation



Upper bound derivation



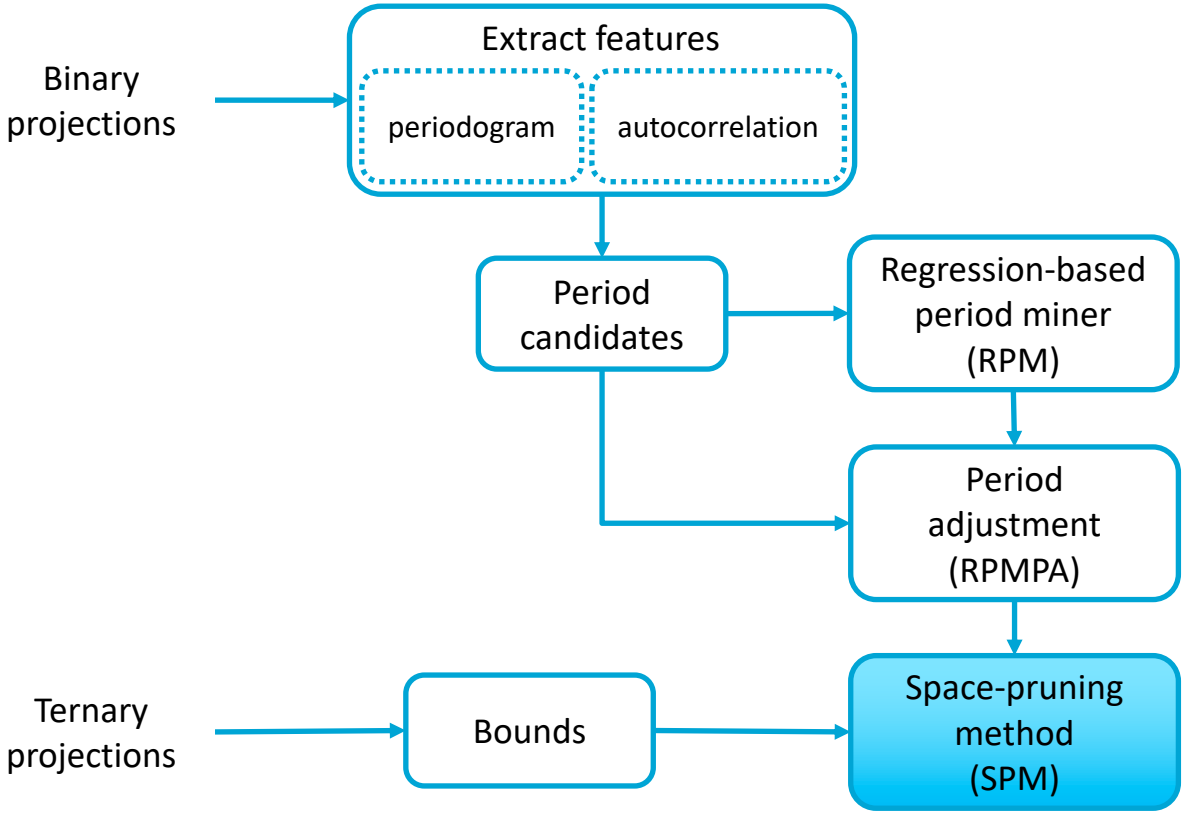
period = difference between the release time of consecutive jobs

$$period_i \leq \min(UI)$$

Q: When is the release time?

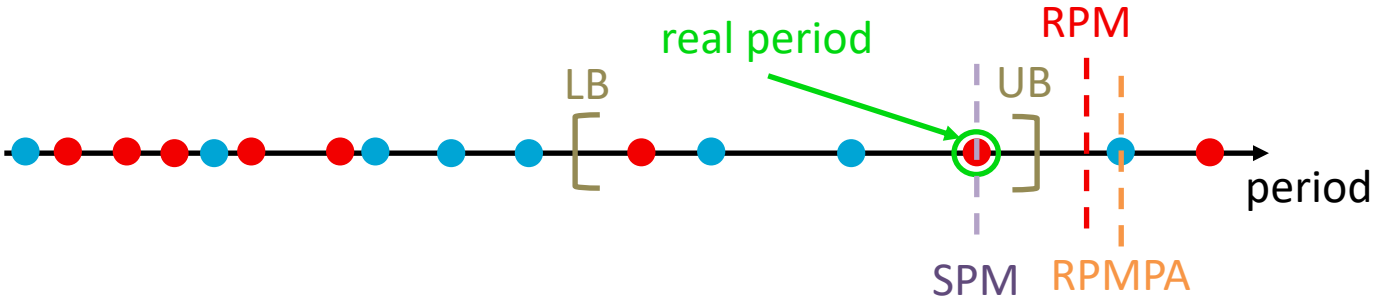
A: We consider extreme cases.

Work in a nutshell



Space pruning method (SPM)

1. Create candidate list from top 20 values of **periodogram** & **autocorrelation**
2. **Filter the candidates based on the bounds**
3. Choose the candidate closest to what RPM estimates



RPM – regression-based period miner
RPMPA – regression-based period miner with period adjustment
UB – upper bound
LB – lower bound

Experiments

What we want to find?

1. Do we have an improvement over the state of the art in terms of

Accuracy

Runtime

2. How do the six considered regression algorithms compare on the two aforementioned criteria?

3. How much does **SPM** improve the accuracy of **RPM** and **RPMPA**?

4. How do our solutions behave under non-ideal conditions?

RPM – regression-based period miner
RPMPA – regression-based period miner with period adjustment
SPM – space pruning method

Error metric

Average estimation error

$$e = \frac{\sum_{i=1}^{N} \frac{\widehat{T}_i - T_i}{T_i}}{N}$$

Estimated period

Actual periods

Number of tasks

Data set

Automotive benchmark application

- Task sets follow Autosar standard
- Periods from {1, 2, 5, 10, 20, 50, 100, 200, 1000}ms
- For simplicity: *automotive traces*

Synthetic task sets

- Random periods between 10ms and 1000ms with a base period of 10ms and log-uniform distribution (commonly used in papers)
- For simplicity: *log-uniform traces*

Simso simulator to generate logs and **extract** traces.

M. Chéramy, P. Hladik, and A. Déplanche. “Simso: A simulation tool to evaluate real-time multiprocessor scheduling algorithms.”, 2014.

Experimental setups

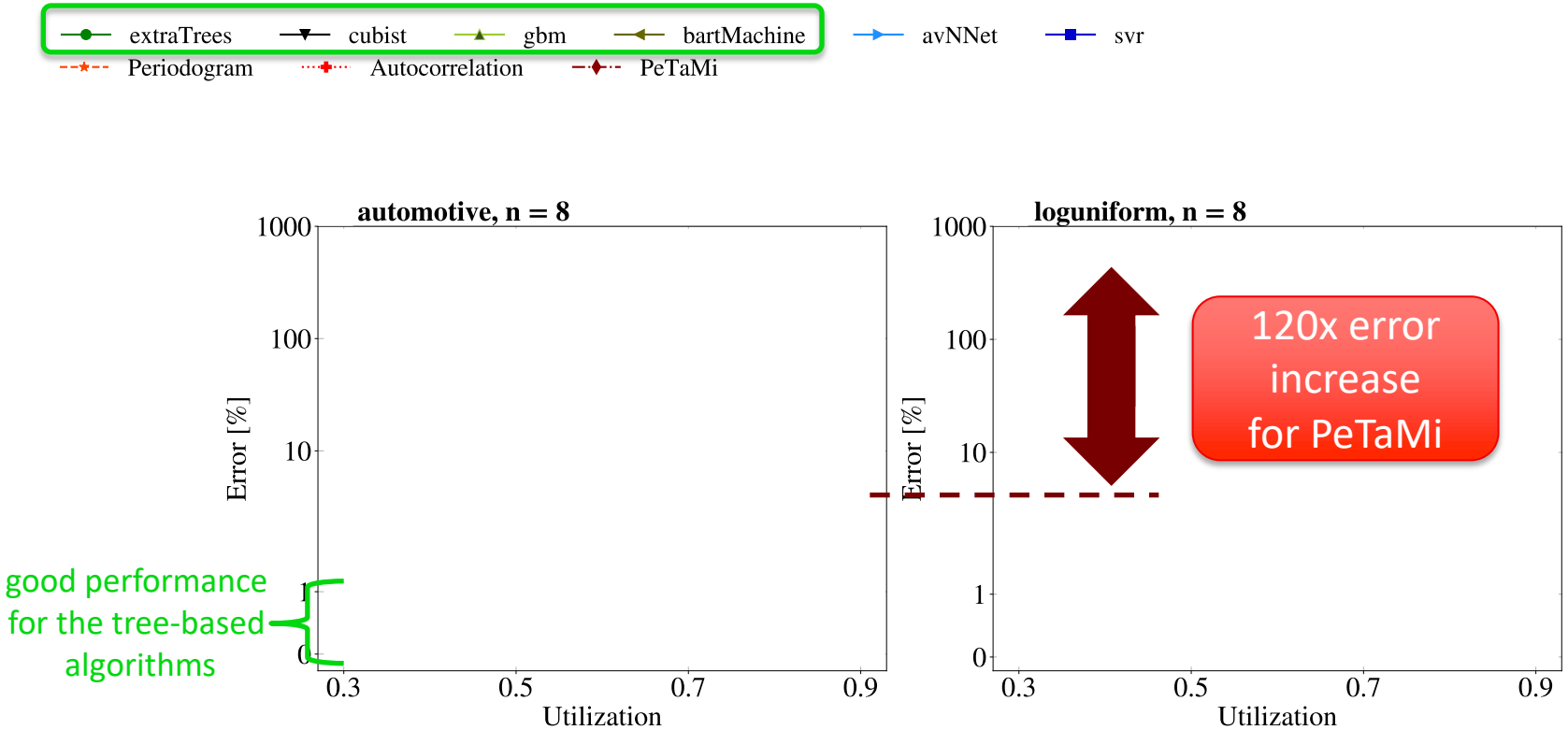
We considered the effects of:

- period ranges
- utilization
- scheduling policy
- release jitter
- high-priority aperiodic tasks
- missing jobs
- execution time variation
- tardiness

RPM
Data set: 2000 traces
Error estimation: cross-validation

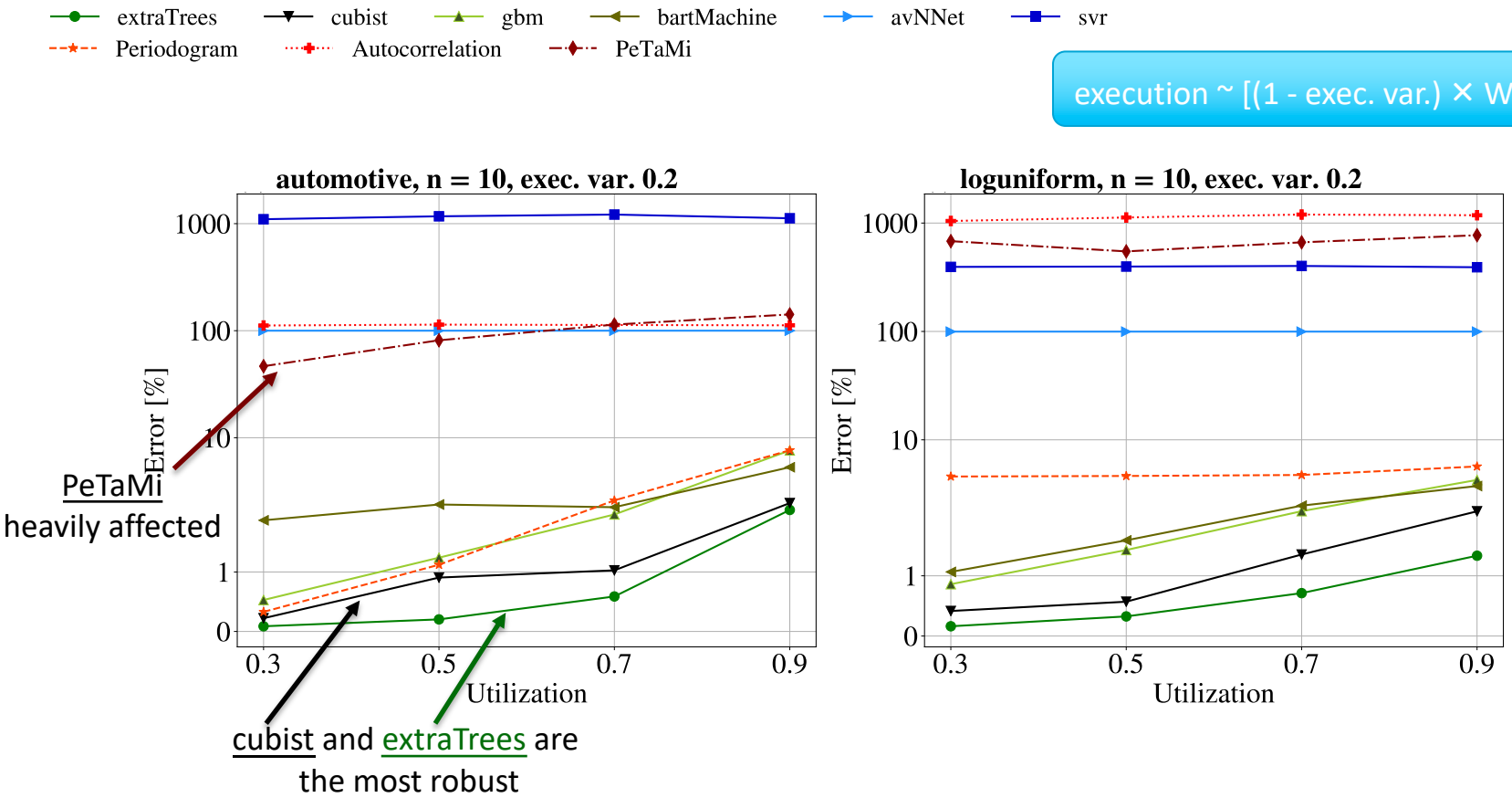
4. How robust are our solutions to sources of non-determinism?

Performance is dependent on the utilization



O. Iegorov, R. Torres, and S. Fischmeister. "Periodic task mining in embedded system traces". *Real-Time and Embedded Technology and Applications Symposium*, pp. 331-340, 2017.

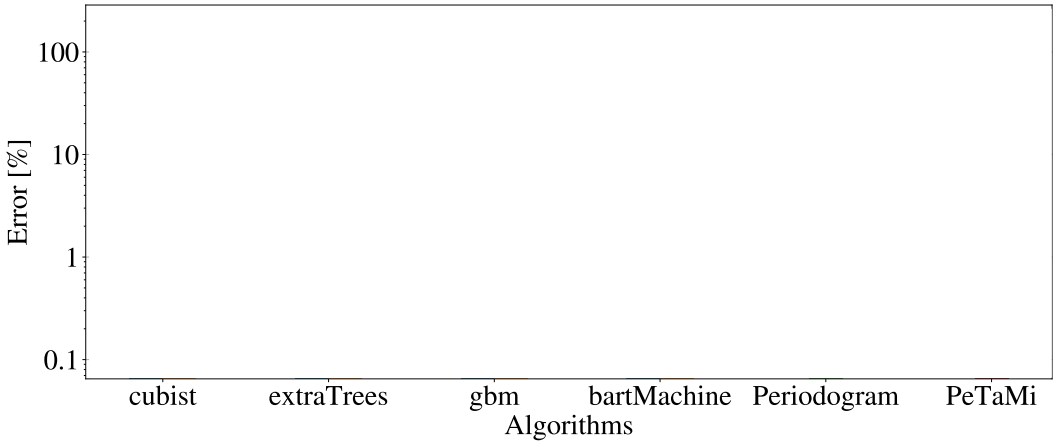
Robustness to variable execution time



Robustness to high-priority aperiodic tasks

- 12 automotive tasks
  - 6 periodic
  - 6 sporadic
- aperiodic tasks following a Poisson process suspending the execution
- task under analysis with <u>medium</u> priority

RPMPA reduces the error for all algorithms



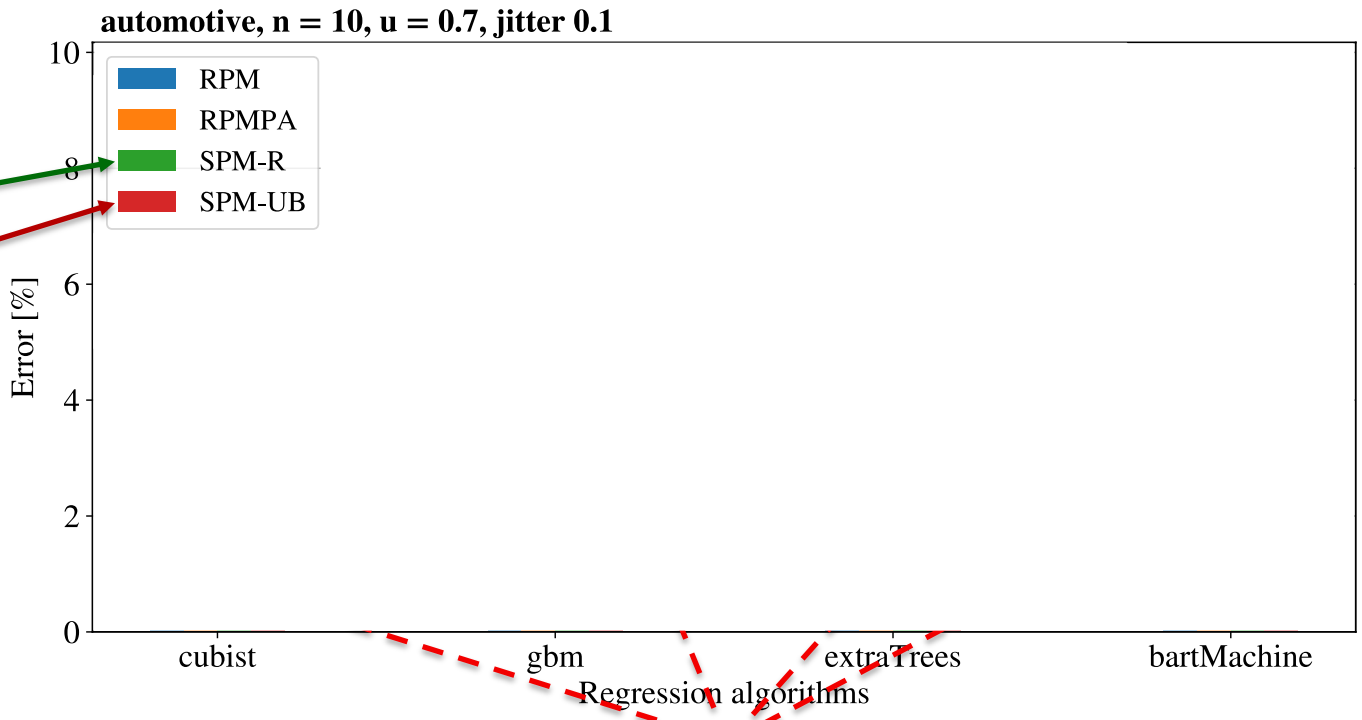
RPM – regression-based period miner
RPMPA – regression-based period miner with period adjustment

Space pruning on jitter

Q: What if no candidate is within the bounds?

A: We choose the prediction from:

- RPM
- upper bound



SPM reduces the error up to **45%**

RPM – regression-based period miner
RPMPA – regression-based period miner with period adjustment
SPM – space pruning method

What is the performance on real data?

Two datasets [4] containing controller area network (CAN) messages obtained from vehicles

We have a lower error than the state of the art

RPM

RPMPA

Data set	Algorithm	RPM [%]	RPMPA [%]
	cubist	3.2461	0.9703
	extraTrees	28.2568	1.6179
	rom	15.8224	1.3919
	lachine	20.8588	14.0103
	TaMi	222.9416	-
	Periodogram	5.3368	-
CAN int	cubist	5.5005	2.8963
		13.0256	1.7039
		19.8341	9.8881
		14.956	5.0264
	PeTaMi	177.6681	-
	Periodogram	9.5448	-

cubist has the **lowest** error

RPMPA significantly reduces the errors

Hacking and Countermeasure Research Lab. http://ocslab.hksecurity. net/Datasets/, 2010. [Online; accessed 01-July-2020].

How about efficiency?

Method	Average runtime [s]
Candidate generation	5.101
RPM (cubist)	2×10^{-6}
PeTaMi	10.731

RPM **2x** faster than PeTaMi

Candidate selection – *O(m)*
m – size of candidate list (40)

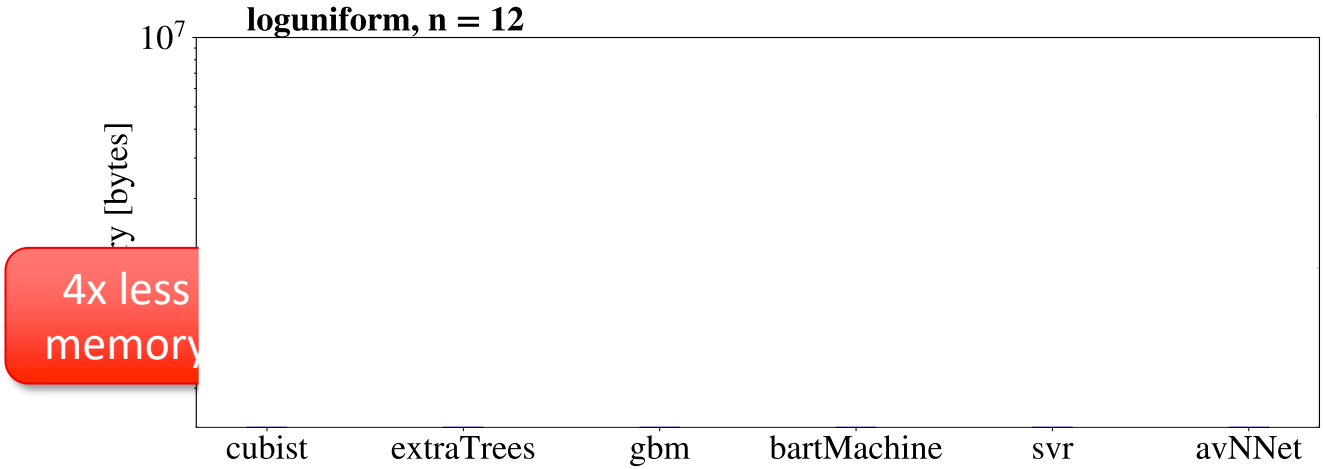
RPMPA **2x** faster than PeTaMi

Candidate generation – *O(n log n)*
Deriving bounds – *O(n)*

SPM comparable to PeTaMi

RPM – regression-based period miner
RPMPA – regression-based period miner with period adjustment
SPM – space pruning method

Memory comparison



Conclusions

1. *Can we use regression-based solutions for the period inference problem?*

- the first RBML method for period inference
  - two to three orders of magnitude **more accurate** than PeTaMi
  - **half the runtime** of PeTaMi

RBML – regression-based machine learning

O. Iegorov, R. Torres, and S. Fischmeister. "Periodic task mining in embedded system traces". *Real-Time and Embedded Technology and Applications Symposium*, pp. 331-340, 2017.

Conclusions

2. *Which regression algorithm would result in a better accuracy for inferring periods?*

- six regression algorithms (cubist, gbm, extraTrees, bartMachine, svr, and avNNet):
  - *cubist* has:
    - **the lowest** memory requirements;
    - **the lowest** runtime;
    - **the lowest** error on real traces.

gbm – gradient boosting
extraTrees – extremely randomized regression trees
bartMachine – Bayesian additive regression trees
svr – support vector regression
avNNet – averaged neural networks

Conclusions

3. *Can we derive a pair of bounds to restrict candidate values?*

- **lower bound** and **upper bound** on period values (space-pruning method).
  - **lowered** the error on traces with jitter up to **45%.**

Conclusions

4. *How robust are our solutions to sources of non-determinism?*

- robustness w.r.t.
  - release time jitter ✓
  - execution time variation ✓
  - presence of high-priority aperiodic tasks ✓
  - missing jobs ✓
  - tardiness ✗
  - scheduling policy ✓

  – *cubist* and *extraTrees* were **robust** against most types of interference.

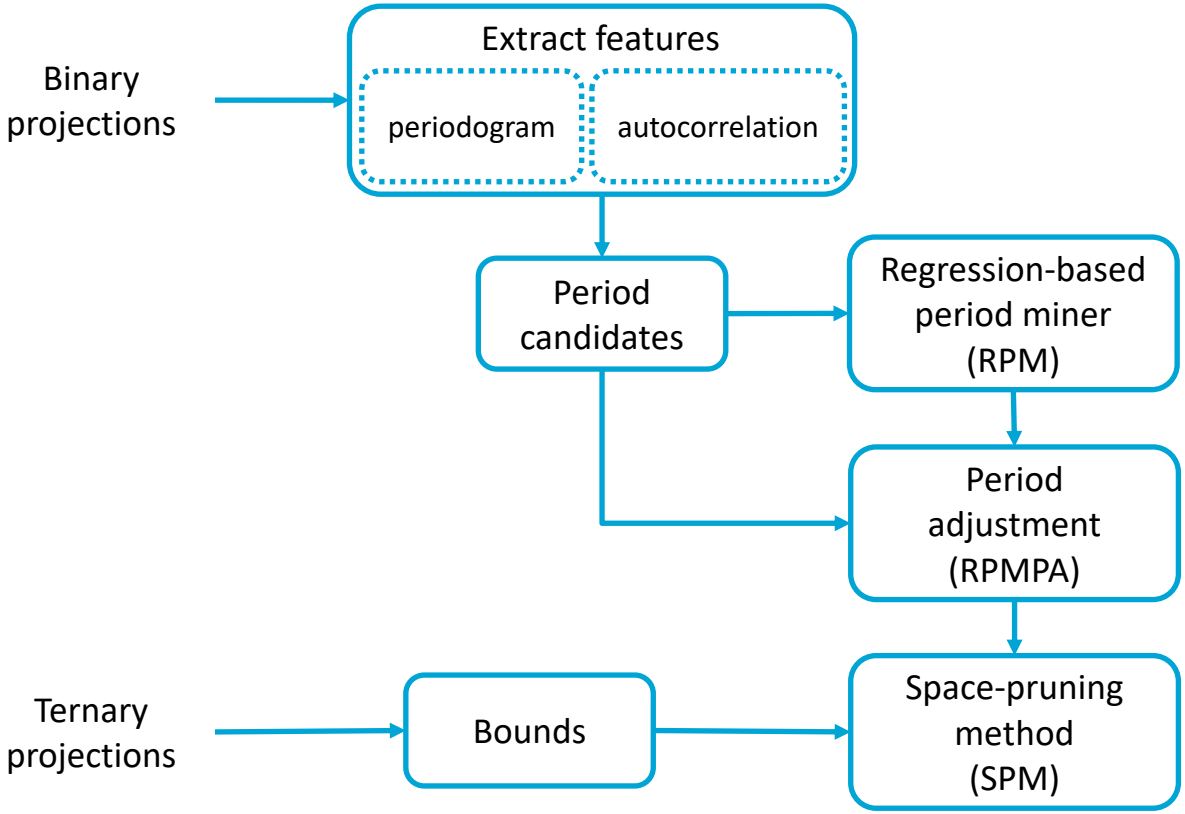
Future work

1. Include information about higher-priority tasks running intervals to derive a **tighter** upper bound.

2. Extending the framework to support **multiprocessor** platforms.

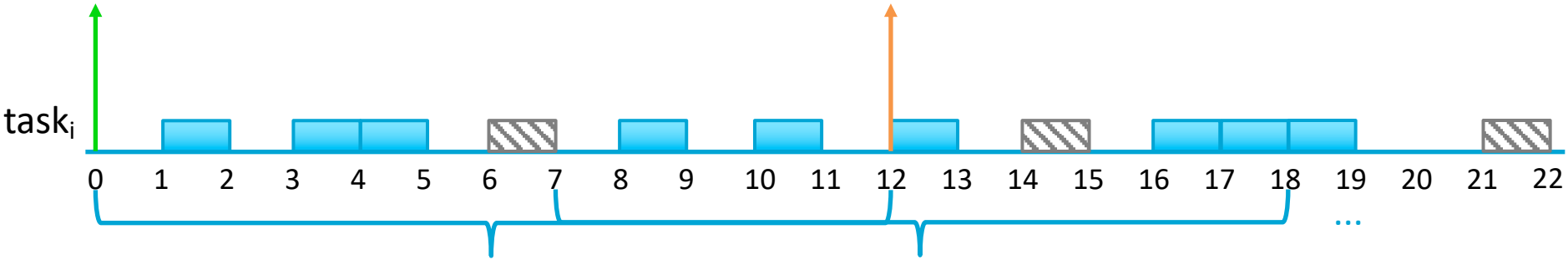
3. Derive the periods of a task **without** requiring an **execution trace**.

Thank you!

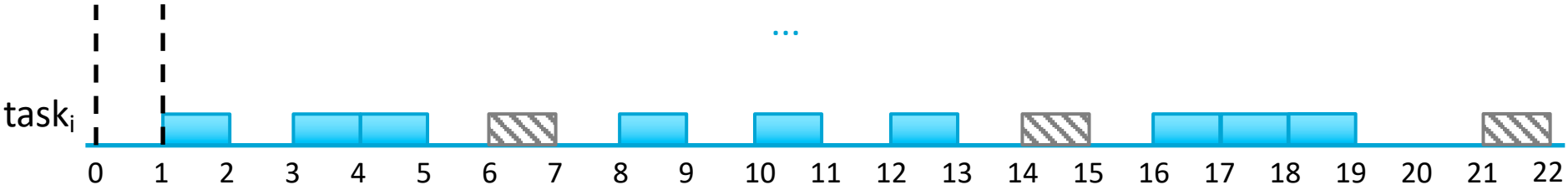

Binary projections
Extract features
periodogram
autocorrelation
Period candidates
Regression-based period miner (RPM)
Period adjustment (RPMPA)
Ternary projections
Bounds
Space-pruning method (SPM)

Extra materials

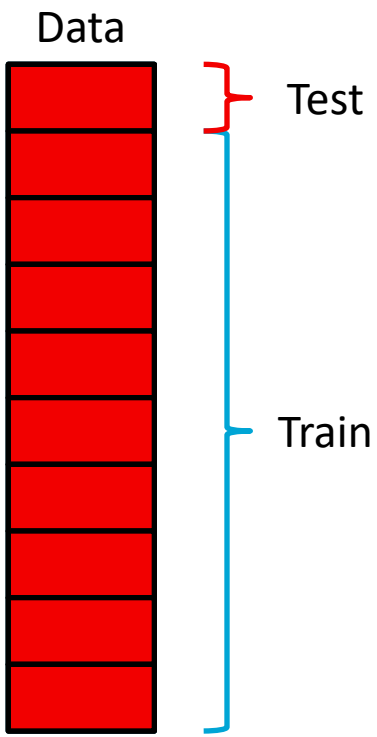
Why SPM is O(n)



Why SPM is O(n)

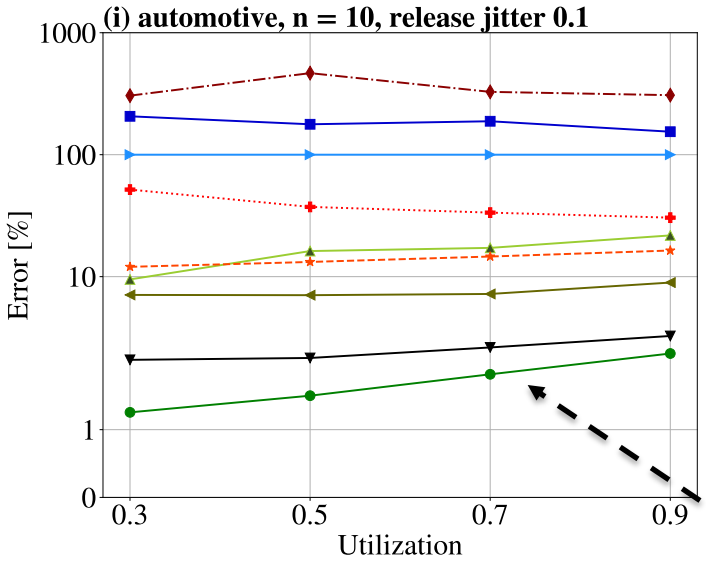


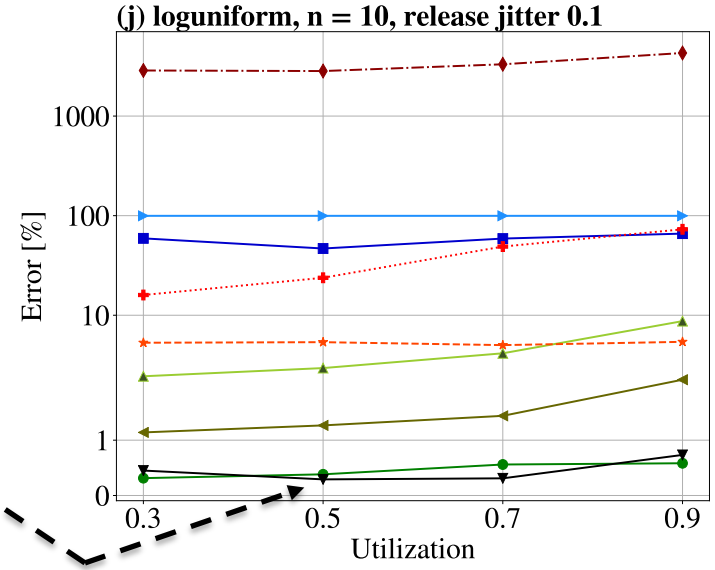
How does cross-validation work?



Robustness to release time jitter

release time jitter
is more challenging than
execution time variation





cubist and extraTrees
still keep an average error **below 2%**