Deriving Timing Properties from System Traces Using Data-driven Techniques

Șerban Vădineanu

Supervisor: Mitra Nasri
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

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

Complexity poses challenges on security

Reverse-engineering
Which properties?

- Functional properties
- Temporal properties

Important for real-time systems
- Periodicity of events
- Periodic activities

Reverse-engineering
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

- **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**

---


What is our aim?

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

---

Work in a nutshell

Binary projections

Extract features

Period candidates

Regression-based period miner (RPM)

Period adjustment (RPMPA)

Space-pruning method (SPM)

Ternary projections

Bounds
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

Extract features
- periodogram
- autocorrelation

Binary projections
- Period candidates
- Regression-based period miner (RPM)
- Period adjustment (RPMPA)
- Space-pruning method (SPM)

Ternary projections
- Bounds
Why extracting features?

- Feature extraction
- Binary projection
- Regression-based machine learning
- Period

• 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

- **Extract features**
  - periodogram
  - autocorrelation

- **Period candidates**
  - Regression-based period miner (RPM)
  - Period adjustment (RPMPA)
  - Space-pruning method (SPM)

- **Binary projections**

- **Ternary projections**

- **Period candidates**

- **Bounds**
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

Extract features
- periodogram
- autocorrelation

Period candidates

Regression-based period miner (RPM)

Period adjustment (RPMPA)

Space-pruning method (SPM)

Binary projections

Ternary projections

Bounds
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?

Which regression algorithms?

<table>
<thead>
<tr>
<th>Tree-based algorithms</th>
<th>Tree-based algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>cubist</td>
<td>extremely randomized regression trees (extraTrees)</td>
</tr>
<tr>
<td>gradient boosting (gbm)</td>
<td>Bayesian additive regression trees (bartMachine)</td>
</tr>
<tr>
<td>support vector regression (svr)</td>
<td>averaged neural networks (avNNet)</td>
</tr>
</tbody>
</table>
What is a regression tree?

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

P1=769, P2=666, P3=5000, A1=4635, A2=10000, A3=5365
Which regression algorithms?

- **cubist**: collapses the tree into a set of rules.
- **extremely randomized regression trees**: trains multiple trees and 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**

\[ y_i = (w \cdot x_i) + b + \epsilon \]

\[ e\text{-deviation} \]

**Averaged neural networks**
Observations

- autocorrelation candidates
- periodogram candidates

often close to the real period
not always accurate

RPM

real period

RPM – regression-based period miner
Work in a nutshell

Extract features
- periodogram
- autocorrelation

Binary projections

Period candidates

Regression-based period miner (RPM)

Period adjustment (RPMPA)

Ternary projections

Bounds

Space-pruning method (SPM)
Period adjustment (RPMPA)

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

2. Choose the candidate closest to what RPM estimates
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

- Binary projections
  - Periodogram
  - Autocorrelation

- Ternary projections
  - Bounds

Extract features

- Period candidates
- Regression-based period miner (RPM)
- Period adjustment (RPMPA)

Space-pruning method (SPM)
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

\[
\text{period}_i > \frac{LNI}{2}
\]

\[
\text{period}_i = 8
\]

\[
\text{execution}_i = 2
\]

\[
\text{deadline}_i = \text{period}_i
\]
Upper bound derivation

since an idle time passed we see an execution

another job started
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.

earliest possible release of previous job

latest possible release of current job

The last time unit of execution before an idle time

perido\_i \leq \min(UI)

Q: When is the release time?
A: We consider extreme cases.
Work in a nutshell

Binary projections

Extract features
- periodogram
- autocorrelation

Period candidates

Regression-based period miner (RPM)

Period adjustment (RPMPA)

Bounds

Space-pruning method (SPM)

Ternary projections
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

\[ e = \frac{\sum_{i=1}^{N} \frac{\hat{T}_i - T_i}{T_i}}{N} \]

- Estimated period
- Actual periods
- Number of tasks

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.

Experimental setups

We considered the effects of:

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

Good performance for the tree-based algorithms

Robustness to variable execution time

- cubist
- extraTrees
- gbm
- bartMachine
- avNNNet
- svr
- Periodogram
- Autocorrelation
- PeTaMi

$$\text{execution} \sim [(1 - \text{exec. var.}) \times \text{WCET}, \text{WCET}]$$

PeTaMi heavily affected

- cubist and extraTrees are the most robust

TU Delft
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 medium priority

RPMPA reduces the error for all algorithms
Space pruning on jitter

Q: What if no candidate is within the bounds?

A: We choose the prediction from:

- RPM
- upper bound

---

**automotive, n = 10, u = 0.7, jitter 0.1**

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

SPM reduces the error up to 45%
What is the performance on real data?

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

<table>
<thead>
<tr>
<th>Data set</th>
<th>Algorithm</th>
<th>RPM [%]</th>
<th>RPMPA [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car hacking</td>
<td>cubist</td>
<td>3.2461</td>
<td>0.9703</td>
</tr>
<tr>
<td></td>
<td>extraTrees</td>
<td>28.2568</td>
<td>1.6179</td>
</tr>
<tr>
<td></td>
<td>gbm</td>
<td>15.8224</td>
<td>1.3919</td>
</tr>
<tr>
<td></td>
<td>bartMachine</td>
<td>20.8588</td>
<td>14.0103</td>
</tr>
<tr>
<td></td>
<td>PeTaMi</td>
<td>222.9416</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Periodogram</td>
<td>5.3368</td>
<td>-</td>
</tr>
<tr>
<td>CAN intrusion</td>
<td>cubist</td>
<td>5.5005</td>
<td>2.8963</td>
</tr>
<tr>
<td></td>
<td>gbm</td>
<td>13.0256</td>
<td>1.7039</td>
</tr>
<tr>
<td></td>
<td>bartMachine</td>
<td>19.8341</td>
<td>9.8881</td>
</tr>
<tr>
<td></td>
<td>PeTaMi</td>
<td>177.6681</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Periodogram</td>
<td>9.5448</td>
<td>-</td>
</tr>
</tbody>
</table>

We have a lower error than the state of the art.

RPM

RPMPA

**cubist has the lowest error**

RPMPA significantly reduces the errors
How about efficiency?

<table>
<thead>
<tr>
<th>Method</th>
<th>Average runtime [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate generation</td>
<td>5.101</td>
</tr>
<tr>
<td>RPM (cubist)</td>
<td>$2 \times 10^{-6}$</td>
</tr>
<tr>
<td>PeTaMi</td>
<td>10.731</td>
</tr>
</tbody>
</table>

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

Candidate generation – $O(n \log n)$

Deriving bounds – $O(n)$

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

loguniform, n = 12

Memory [bytes]

10^7

4x less memory

cubist  extraTrees  gbm  bartMachine  svr  avNNNet
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
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
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.
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.
Extra materials
Why SPM is O(n)
Why SPM is $O(n)$

![Diagram showing tasks and timing](image-url)
How does cross-validation work?
Robustness to release time jitter

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

Release time jitter is more challenging than execution time variation