Early WCET Prediction using Machine Learning
(Work in Progress)

Armelle Bonenfant\textsuperscript{1}, Denis Claraz\textsuperscript{2}, Marianne de Michiel\textsuperscript{1} and Pascal Sotin\textsuperscript{1}

\textsuperscript{1}IRIT\hspace{1cm}\textsuperscript{2}Continental

Toulouse, France

June 27, 2017
WCET Workshop
Dubrovnik, Croatia
### Cost of a Bug

Relative cost of a bug (according to IBM)

<table>
<thead>
<tr>
<th>Phase</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design phase</td>
<td>1</td>
</tr>
<tr>
<td>After development</td>
<td>5</td>
</tr>
<tr>
<td>After deployment</td>
<td>100</td>
</tr>
</tbody>
</table>

‘The cost to fix an error found after product release was four to five times as much as one uncovered during design, and up to 100 times more than one identified in the maintenance phase.’

— IBM Systems Sciences Institute
Cost of a Bug

Relative cost of a bug (according to IBM)

<table>
<thead>
<tr>
<th>Phase</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design phase</td>
<td>1</td>
</tr>
<tr>
<td>After development</td>
<td>5</td>
</tr>
<tr>
<td>After deployment</td>
<td>100</td>
</tr>
</tbody>
</table>

‘The cost to fix an error found after product release was four to five times as much as one uncovered during design, and up to 100 times more than one identified in the maintenance phase.’

— IBM Systems Sciences Institute

→ Transfer to WCET?
What the Industry Wants

- Source code
- Some magic tool
- Compiler options
- Target architecture

WCET prediction

Pessimistic/accurate

→

"This is a very wild idea and I doubt that it will work on real programs."
— An anonymous reviewer

Pascal Sotin – Early WCET
What the Industry Wants

source code → some magic tool

+ compiler options
+ target architecture

→ WCET prediction

pessimistic/accurate

→ Realistic expectations?

‘This is a very wild idea and I doubt that it will work on real programs.’

— An anonymous reviewer
## Preliminary Results Preview

<table>
<thead>
<tr>
<th>Technique</th>
<th>Accuracy</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training set</td>
<td>Evaluation set</td>
<td></td>
</tr>
<tr>
<td><strong>Multi-Layer Perceptron</strong></td>
<td>Err. 10%</td>
<td>Err. 11%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24% u.</td>
<td>24% u.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0% o.</td>
<td>1% o.</td>
<td></td>
</tr>
<tr>
<td><strong>Random Forest</strong></td>
<td>Err. 4%</td>
<td>Err. 12%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3% u.</td>
<td>4% u.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0% o.</td>
<td>2% o.</td>
<td></td>
</tr>
<tr>
<td><strong>Linear Regression</strong></td>
<td>Err. 10%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8% u.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4% o.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

→ So far the results on TACLeBench are partial and disappointing

These results are due to Frédéric Fort.
Our Proposal: Use Machine Learning

**Machine Learning Input: a Spreadsheet**

<table>
<thead>
<tr>
<th>Program</th>
<th>Characteristic 1</th>
<th>Characteristic 2</th>
<th>⋮</th>
<th>WCET</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td>1410</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td>6912</td>
</tr>
<tr>
<td>⋮</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Machine Learning Output: a Formula**

\[ WCET \sim f(\text{characteristics}) \]
Outline of the Presentation

1. Introduction
2. Learning Framework
3. Source Code Analysis
4. Preliminary Results
Learning Framework

- Source code analysis
- Preliminary results
- Conclusion

**Learning Framework**

- Compiler
- Source code
- Timing analysis
- WCET estimate
- Machine Learning
- WEKA
- Prediction error
- WCET prediction

**Formulas**

- C1
- C2
- WCET
- A
- B

Pascal Sotin – Early WCET
### Issue 1/3: The Learning Set

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C1</th>
<th>C2</th>
<th>...</th>
<th>WCET</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Machine Learning requires:**

- Large sets (more than 1 000 programs)
- Representative sets

**Candidates program sets**

- Benchmarks: good repr., border size, availability issue
- Industrial bank of functions: unlimited size, doubtful repr., compilation optim. issue
- Generated programs:repr. ok, size not ok

---

Pascal Sotin – Early WCET
Issue 2/3: The Characteristics

Machine Learning requires:

- Numerical or discrete characteristics
- Characteristics controlling the learnt attribute
  - Syntactic characteristics are insufficient

We rely on worst-case numerical metrics (details follow shortly)
Do it **exists** a set of characteristics of the source code controlling the WCET?

→ We consider prediction in $\pm 20\%$ as OK
Outline of the Presentation

1. Introduction
2. Learning Framework
3. Source Code Analysis
4. Preliminary Results
Worst Case Event Count Analysis

The worst case event count analysis delivers an over-approximation of the number of events by category triggered by any symbolic execution of the source program.

- Example of categories:
  - Reading a variable
  - Performing a non-trivial multiplication
  - “Branching” back to the loop head

- A mapping from category to number is called a “metric”
Playing with Metrics

- Sequential code gives one metric
- Alternatives should be combined with care
  - $m_1 \sqcup m_2$ is a sound but coarse approximation of “$m_1$ or $m_2$”

$$\begin{bmatrix}
Addition \mapsto 3 \\
Product \mapsto 4 
\end{bmatrix} \sqcup \begin{bmatrix}
Addition \mapsto 4 \\
Division \mapsto 2
\end{bmatrix} = \begin{bmatrix}
Addition \mapsto 4 \\
Product \mapsto 4 \\
Division \mapsto 2
\end{bmatrix}$$

- If neither $m_1 \prec m_2$ nor $m_2 \prec m_1$ we continue with $\{m_1, m_2\}$
  - The analysis is disjunctive (complexity issue)
  - Using bounds on the categories costs, we strengthen the $\prec$ relation

$$\begin{bmatrix}
Jump \mapsto 11 \\
Addition \mapsto 2
\end{bmatrix} \prec \begin{bmatrix}
Jump \mapsto 14 \\
Product \mapsto 5
\end{bmatrix}$$

- Loop bounds are needed
- Eventually $\sqcup$ is used to get a unique metric
Our implementation of the analysis considers:

<table>
<thead>
<tr>
<th>Family</th>
<th>Category</th>
<th>Optimistic</th>
<th>Pessimistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>Simple</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Multiplication</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Division</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Control</td>
<td>Unconditional branch</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Conditional branch</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Computed branch</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Call</td>
<td>0.5</td>
<td>10</td>
</tr>
<tr>
<td>Memory</td>
<td>Address setting</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Load</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Store</td>
<td>2</td>
<td>20</td>
</tr>
</tbody>
</table>
Outline of the Presentation

1. Introduction
2. Learning Framework
3. Source Code Analysis
4. Preliminary Results
Experiments Setting

- Programs are compiled with gcc
  - The target is ARM
- Worst Case Event Count analysis is performed by oRange
- WCET analysis is performed by OTAWA
  - The model is a simple ARMv5
Program Sets Involved

- **Training set (10 000)**
  - Generated randomly (with if and for statements)

- **Evaluation set (5 000)**
  - Generated the same way

- **TACLeBench (23)**
  - Sequential part of the benchmark
Learning Techniques Explored

- **Multi-Layer Perceptron**
  - State-of-the-art neural network
- **Random Forest**
  - Huge decision tree
- **Linear Regression**
  - Best linear combination of the characteristics
Applying the learnt formula on a set of programs gives statistics

- Correlation coefficient: 0.9961
- Mean absolute error: 187.9878
- Root mean squared error: 490.4514
- Relative absolute error: 4.2395
- Root relative squared error: 8.972
- Total Number of Instances: 10000
- Underestimations: 283
- Overestimations: 0

We make the following summaries:

<table>
<thead>
<tr>
<th>Err.</th>
<th>4%</th>
</tr>
</thead>
<tbody>
<tr>
<td>3% u.</td>
<td>0% o.</td>
</tr>
</tbody>
</table>
## Current Accuracy of our Approach

<table>
<thead>
<tr>
<th>Technique</th>
<th>Accuracy</th>
<th>Training set</th>
<th>Evaluation set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Layer Perceptron</td>
<td>Err. 10%</td>
<td>24% u.</td>
<td>24% u.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0% o.</td>
<td>1% o.</td>
</tr>
<tr>
<td>Random Forest</td>
<td>Err. 4%</td>
<td>3% u.</td>
<td>4% u.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0% o.</td>
<td>2% o.</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>Err. 10%</td>
<td>8% u.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4% o.</td>
<td></td>
</tr>
</tbody>
</table>

→ So far the results on TACLeBench are partial and disappointing

These results are due to Frédéric Fort.
Linear Regression Formula

$$\text{WCET} = 0.0506 \times \text{SimpleOp} + 2.2557 \times \text{Mult} + 0 \times \text{Div} + 1.0391 \times \text{CondBr} + 2.638 \times \text{UncondBr} + 0 \times \text{CalcBr} + 0 \times \text{Call} + 0 \times \text{Return} + 1.9445 \times \text{Address} + 0 \times \text{Load} + 0 \times \text{Store} + 18.4446$$
Related Work

Gustafsson et al.\(^1\),\(^2\) follow the **same objective with different tools**

- **Measurement-based approach**
  - No need for hardware model but need the hardware itself

- **Ad-hoc learning techniques**

- **Evaluation on the Mälardalen benchmark gives:**
  
  - **Accurate approach:**
    - Err. 8%
    - 4% u. | 9% o.

  - **Pessimistic approach:**
    - Err. 31%
    - 0% u. | 52% o.


We seek early WCET predictions through Machine Learning

We proposed a source analysis for retrieving program metrics
  → Required for both learning and predicting

Several question are opened
  • Relevant characteristics
  • Best learning technique