Refinement of Worst-Case Execution Time Bounds by Graph Pruning

Florian Brandner
Unité d’Informatique et d’Ing. des Systèmes
ENSTA-ParisTech

Alexander Jordan
Embedded Systems Engineering Sect.
Technical University of Denmark

This work is partially supported by the EC project T-CREST.
Real-Time Systems

Strict timing guarantees

- Critical tasks have to be completed in time
Real-Time Systems

Strict timing guarantees

- Critical tasks have to be completed in time
- Bound *Worst-Case Execution Time* (WCET)

![Graph showing execution times with worst-case execution time bound and overestimation.](image-url)
WCET Analysis (1)

Three analysis phases:

(1) Loop bounds & flow facts
(2) Pipeline & caches
(3) Longest path search

(IPET)
WCET Analysis (1)

Three analysis phases:

(1) Loop bounds & flow facts
WCET Analysis (1)

Three analysis phases:
(1) Loop bounds & flow facts
(2) Pipeline & caches
WCET Analysis (1)

Three analysis phases:

(1) Loop bounds & flow facts

(2) Pipeline & caches

(3) Longest path search (IPET)

WCET = 2 + 7 + 10 \cdot 5 + 1 = 60
WCET Analysis (2)

Bound longest possible execution time of a program
- Covering all potential execution paths
- Covering all potential program inputs
- Covering all potential hardware states
WCET Analysis (2)

Bound longest possible execution time of a program
• Covering all potential execution paths
• Covering all potential program inputs
• Covering all potential hardware states

A priori all executions are equally considered relevant
Criticalities

\[ W(BB0) = 2 \]
\[ W(BB1) = 3 \]
\[ W(BB2) = 7 \]
\[ W(BB3) = 1 \]
\[ W(BB4) = 5 \]
\[ W(BB5) = 1 \]

\[ x := 10 \]
\[ x := 7 \]
\[ x < 10 \]

Criticality:

- WCET(\(BBn\)): Longest path over \(BBn\).
- WCET: Longest path in the graph (from \(r\) to \(t\))
- \(\text{Crit}(BBn) = \frac{\text{WCET}(BBn)}{\text{WCET}}\)
Criticalities

Criticality:

- $\text{WCET}(BB_n)$: Longest path over $BB_n$.
- $\text{WCET}$: Longest path in the graph (from $r$ to $t$)
- $\text{Crit}(BB_n) = \frac{\text{WCET}(BB_n)}{\text{WCET}}$

$\text{Crit}(BB3) = \frac{7}{60} = 0.12$
$\text{Crit}(BB1) = \frac{56}{60} = 0.93$
## Criticality Distribution: Debie1

<table>
<thead>
<tr>
<th>Problem</th>
<th>BBs</th>
<th>$l_0$</th>
<th>$l_1$</th>
<th>$l_2$</th>
<th>$l_3$</th>
<th>$l_4$</th>
<th>$l_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>debie1</td>
<td>83</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>13</td>
<td>19</td>
<td>45</td>
</tr>
<tr>
<td>debie3a</td>
<td>74</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>57</td>
</tr>
<tr>
<td>debie3b</td>
<td>74</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>59</td>
</tr>
<tr>
<td>debie3c</td>
<td>74</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>59</td>
</tr>
<tr>
<td>debie4a</td>
<td>285</td>
<td>31</td>
<td>192</td>
<td>0</td>
<td>19</td>
<td>3</td>
<td>40</td>
</tr>
<tr>
<td>debie4b</td>
<td>285</td>
<td>236</td>
<td>3</td>
<td>14</td>
<td>0</td>
<td>3</td>
<td>29</td>
</tr>
<tr>
<td>debie4c</td>
<td>285</td>
<td>260</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>debie4d</td>
<td>285</td>
<td>264</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>debie5a</td>
<td>138</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>120</td>
</tr>
<tr>
<td>debie5b</td>
<td>138</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>132</td>
</tr>
<tr>
<td>debie6a</td>
<td>376</td>
<td>53</td>
<td>24</td>
<td>2</td>
<td>105</td>
<td>0</td>
<td>192</td>
</tr>
<tr>
<td>debie6b</td>
<td>376</td>
<td>52</td>
<td>22</td>
<td>4</td>
<td>106</td>
<td>0</td>
<td>192</td>
</tr>
<tr>
<td>debie6c</td>
<td>376</td>
<td>52</td>
<td>22</td>
<td>143</td>
<td>4</td>
<td>0</td>
<td>155</td>
</tr>
<tr>
<td>debie6d</td>
<td>376</td>
<td>12</td>
<td>24</td>
<td>2</td>
<td>0</td>
<td>144</td>
<td>194</td>
</tr>
</tbody>
</table>

*Intervals: $0 \leq l_0 < 0.25 < l_1 < 0.5 < l_2 < 0.75 < l_3 < 0.9 < l_4 < 0.99 < l_5 \leq 1$
Iterative Graph Pruning

Improving WCET bounds

• Many basic blocks turn out to be *uncritical*
Iterative Graph Pruning

Improving WCET bounds

- Many basic blocks turn out to be *uncritical*
- Why do we then analyze them?
Iterative Graph Pruning

Improving WCET bounds

- Many basic blocks turn out to be *uncritical*
- Why do we then analyze them?
- Can we remove uncritical blocks?
  - Focus on relevant code only
  - More precise WCET
  - Faster analysis
Iterative Graph Pruning: Example

\[ r \]

\[ W(BB0) = 2 \]
\[ x := 10 \]

\[ W(BB1) = 3 \]
\[ x := 7 \]
\[ < x < 10 \]

\[ W(BB2) = 7 \]
\[ < x < 10 \]

\[ W(BB3) = 1 \]

\[ W(BB4) = 5 \]

\[ W(BB5) = 1 \]
Iterative Graph Pruning: Example

\[ W(BB0) = 2 \]
\[ W(BB1) = 3 \]
\[ W(BB2) = 7 \]
\[ W(BB3) = 1 \]
\[ W(BB4) = 5 \]
\[ W(BB5) = 1 \]

\[ x := 10 \]
\[ x := 7 \]

\[ < x < 10 \]

\[ S_0 \cup S_1 = 60 \]
\[ S_0 = BB0, BB2, BB4, BB5 \]
\[ S_1 = BB1 \]
\[ S_2 = BB3 \]
Iterative Graph Pruning: Example

\[ W(BB0) = 2 \]
\[ W(BB1) = 3 \]
\[ W(BB2) = 7 \]
\[ W(BB3) = 1 \]
\[ W(BB4) = 5 \]
\[ W(BB5) = 1 \]

\[ x := 10 \]
\[ x := 7 \]
\[ \langle x < 10 \rangle \]

\[ WCET = 2 + 7 + 7 \cdot 4 + 1 = 38 \]
Iterative Graph Pruning: Example

\[ W(\text{BB0}) = 2 \]
\[ W(\text{BB1}) = 3 \]
\[ W(\text{BB2}) = 7 \]
\[ W(\text{BB3}) = 1 \]
\[ W(\text{BB4}) = 5 \]
\[ W(\text{BB5}) = 1 \]

\[ x := 10 \]
\[ x := 7 \]

\[ < x < 10 \]
\[ 7 \]

\[ \text{WCET} = 2 + 7 + 7 \cdot 4 + 1 = 38 \]
Iterative Graph Pruning: Example

\[ W(BB0) = 2 \]
\[ W(BB1) = 3 \]
\[ W(BB2) = 7 \]
\[ W(BB3) = 1 \]
\[ W(BB4) = 5 \]
\[ W(BB5) = 1 \]

\[ x := 10 \]
\[ x := 7 \]
\[ < x < 10 \]
\[ 7 \]

\[ WCET = 2 + 7 + 10 \cdot 5 + 1 = 60 \]
Iterative Graph Pruning: Example

\[ W(BB0) = 2 \]
\[ W(BB1) = 3 \]
\[ W(BB2) = 7 \]
\[ W(BB3) = 1 \]
\[ W(BB4) = 5 \]
\[ W(BB5) = 1 \]
\[ x := 10 \]
\[ x := 7 \]
\[ < x < 10 \]
\[ WCET = 2 + 3 + 10 \cdot 5 + 1 = 56 \]

\[ S_0 \cup S_1 = \{ BB0, BB1, BB2, BB4, BB5 \} \]
\[ S_2 = \{ BB3 \} \]
Iterative Graph Pruning: Example

\[ W(BB0) = 2 \]
\[ W(BB1) = 3 \]
\[ W(BB2) = 7 \]
\[ W(BB3) = 1 \]
\[ W(BB4) = 5 \]
\[ W(BB5) = 1 \]

\[ x := 10 \]
\[ x := 7 \]
\[ < x < 10 \]

\[ S_0 \cup S_1 \]

\[ \text{BB0, BB1, BB2, BB4, BB5} \]

\[ \text{BB3} \]

\[ S_2 \]
Iterative Graph Pruning: Algorithm

**Input:** $G = (V, E)$ The program’s control-flow graph  
$S_0, \ldots, S_n$ Block sets sorted by path length

1: $u_{\text{WCET}} := 0$
2: for $i = 1$ to $n$
3: if $u_{\text{WCET}} \geq \text{pathlength}(S_i)$
4: return $u_{\text{WCET}}$
5: let $V' = S_0 \cup \ldots \cup S_i$, $G' = (V', E \cap V' \times V')$ in
6: $u_{\text{WCET}} := \max(u_{\text{WCET}}, \text{WCEToverAny}(G', S_i))$
7: return $u_{\text{WCET}}$
Fast vs. Precise WCET Analysis

<table>
<thead>
<tr>
<th>Two-Stage WCET analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Combine fast with precise analysis</td>
</tr>
<tr>
<td>• Fast analysis</td>
</tr>
<tr>
<td>• Compute block sets</td>
</tr>
<tr>
<td>• Check WCET increase while iterating</td>
</tr>
<tr>
<td>• Precise analysis to verify</td>
</tr>
</tbody>
</table>
Fast vs. Precise WCET Analysis

Two-Stage WCET analysis
- Combine fast with precise analysis
- Fast analysis
  - Compute block sets
  - Check WCET increase while iterating
- Precise analysis to verify

Non-Iterative Pruning
- Heuristically construct a pruned graph
  - Using Criticality?
  - Using Criticality estimates?
- Apply precise analysis to pruned graph
# Experiments

## Setup

- Commercial WCET analysis tool\(^a\) (AbsInt aiT, 12.10)
- Freescale mpc5554 and mpc755s (PowerPC)

\(^a\) [http://www.absint.com/ait/](http://www.absint.com/ait/)

Experiments

Setup

• Commercial WCET analysis tool\(^a\) (AbsInt aiT, 12.10)
  • Freescale mpc5554 and mpc755s (PowerPC)

• Two real-time benchmarks
  • Debie1: satellite instrument control
  • Papabench: flight control
Experiments

<table>
<thead>
<tr>
<th>Setup</th>
</tr>
</thead>
</table>
| • Commercial WCET analysis tool\(^a\) (AbsInt aiT, 12.10)  
• Freescale mpc5554 and mpc755s (PowerPC) |
| • Two real-time benchmarks  
  • Debie1: satellite instrument control  
  • Papabenche: flight control |
| • 28 analysis problems\(^b\) |

\(^a\)http://www.absint.com/ait/  
\(^b\)http://www.mrtc.mdh.se/projects/WCC/2011/
WCET Reductions (mpc5554)

WCET Reductions up to 6%.

It is usually already close to measured bounds.
WCET Reductions (mpc755s)

WCET Reductions up to 12%.
Iterations of f1a: WCET (mpc5554)

- WCET bound (original)
- $ub_{wcet}$ (IGP)
- $WCET_i$ (IGP)
Iterations of f1a: Problem Size (mpc5554)
Conclusion

• Criticality
  • Novel compiler-centric metric
  • Proved interesting for WCET analysis
  • Cheap yet accurate estimation
Conclusion

• Criticality
  • Novel compiler-centric metric
  • Proved interesting for WCET analysis
  • Cheap yet accurate estimation

• Iterative Graph Pruning
  • Based on Criticality
  • Allows elimination of uncritical code
  • Successfully reduces overestimation
  • Causes quite some overhead (9x on average)
    • Proof-of-Concept implementation
    • Treats WCET analysis as black box
    • Incremental analysis techniques needed