Combinatorial Abstraction Refinement for Feasibility Analysis

Martin Stigge

Uppsala University, Sweden

Joint work with Wang Yi
Problem Overview

Workload Model
- Task A
- Task B
- Task C

Scheduler Model
- EDF/Static Prio/...

Our Setting:
- DRT tasks
- Static Priorities
- Precise Test
Problem Overview

Workload Model

- High priority: Tasks A
- Medium priority: Tasks B
- Low priority: Task C

Scheduler Model

- EDF/Static Prio/...

Our Setting:
- DRT tasks
- Static Priorities
- Precise Test
The Digraph Real-Time (DRT) Task Model
(S. et al, RTAS 2011)

- Generalizes periodic, sporadic, GMF, RRT, ... 
- *Directed graph* for each task
  - Vertices $J$: jobs to be released (with WCET and deadline)
  - Edges $(J_i, J_j)$: minimum inter-release delays $p(J_i, J_j)$
DRT: Semantics

Path \( \pi = (J_1) \)

Path \( \pi = (J_1, J_2) \)

Path \( \pi = (J_1, J_2, J_3) \)

\( t_{0} = 6, t_{10} = 10, t_{25} = 25, t_{1} = 10, t_{12} = 12, t_{13} = 13, t_{25} = 25, t_{54} = 13, t_{44} = 13, t_{50} = 29, t_{1} = 10, t_{10} = 10 \)

Martin Stigge

Combinatorial Abstraction Refinement
DRT: Semantics

Path \( \pi = (J_1) \)
Path $\pi = (J_1, J_2)$
Path $\pi = (J_1, J_2, J_3)$
Complexity Results for DRT Schedulability

EDF
- \textit{Pseudo-polynomial}
- Dbf-based analysis \cite{RTAS2011}
- Equivalent to Feasibility

Static Priorities
- Strongly coNP-hard
- Already for trees or cycles \cite{ECRTS2012}
- Efficient solution?
## Complexity Results for DRT Schedulability

### EDF
- *Pseudo-polynomial*
- Dbf-based analysis
  - [RTAS 2011]
- Equivalent to Feasibility

### Static Priorities
- Strongly *coNP-hard*
- Already for trees or cycles
  - [ECRTS 2012]
- Efficient solution?
1 Problem Introduction
   - Digraph Real-Time Tasks
   - Complexity Results

2 Analysis Approach
   - Request Functions
   - Rf-based Test

3 Combinatorial Abstraction Refinement
   - Abstraction Trees
   - Refinement Procedure

4 Evaluation
Problem Introduction
- Digraph Real-Time Tasks
- Complexity Results

Analysis Approach
- Request Functions
- Rf-based Test

Combinatorial Abstraction Refinement
- Abstraction Trees
- Refinement Procedure

Evaluation
Testing the Scheduling Window

High priority

Medium priority

Low priority

Is C schedulable?

Scheduling window of C
Testing the Scheduling Window

High priority

Medium priority

Low priority

Is C schedulable?

Scheduling window of C
Request Functions

\[ J_1 \langle 6, 10 \rangle \]
\[ J_2 \langle 5, 25 \rangle \]
\[ J_3 \langle 1, 10 \rangle \]
\[ J_4 \langle 2, 12 \rangle \]
\[ J_5 \langle 10, 50 \rangle \]

\[ rf(t) \]

\[ rf(J_1, J_2, J_3) \]
Lemma

A job $J$ is schedulable iff for all combinations of request functions $rf(T)$ of higher priority tasks:

$$\exists t \leq d(J) : e(J) + \sum_{T \in \tau} rf(T)(t) \leq t.$$  (1)
Request Functions: Schedulability Test

Lemma

A job $J$ is schedulable iff for all combinations of request functions $rf^{(T)}$ of higher priority tasks:

$$\exists t \leq d(J): e(J) + \sum_{T \in \tau} rf^{(T)}(t) \leq t.$$  \hfill (1)
Lemma

A job $J$ is schedulable iff for all combinations of request functions $rf(T)$ of higher priority tasks:

$$\exists t \leq d(J): e(J) + \sum_{T \in \tau} rf(T)(t) \leq t.$$  \hspace{1cm} (1)

Problem: Naive test double exponential!

1. Number of paths per task
2. Number of combinations
Request Functions: Domination

\[ rf(t) \]

- \( J_1 \langle 6, 10 \rangle \)
- \( J_2 \langle 5, 25 \rangle \)
- \( J_3 \langle 1, 10 \rangle \)
- \( J_4 \langle 2, 12 \rangle \)
- \( J_5 \langle 10, 50 \rangle \)

\[ rf(J_1, J_2, J_3) \]

Graph showing the relationships between the functions and their values at specific time points.

Martin Stigge
Combinatorial Abstraction Refinement
Request Functions: Domination

\[
\begin{align*}
J_1 &\langle 6, 10 \rangle \\
J_2 &\langle 5, 25 \rangle \\
J_3 &\langle 1, 10 \rangle \\
J_4 &\langle 2, 12 \rangle \\
J_5 &\langle 10, 50 \rangle \\
\end{align*}
\]

\[
\text{rf}(t)
\]

\[
\text{rf}(J_1, J_2, J_3)
\]

\[
\text{rf}(J_3, J_4, J_2)
\]
Request Functions: Domination

\[ \text{rf}(t) \]

\[ rf(J_1, J_2, J_3) \]

\[ rf(J_3, J_4, J_2) \]
Combinatorial Explosion

Lemma

A job $J$ is schedulable if for all combinations of request functions $rf^{(T)}$ of higher priority tasks:

$$\exists t \leq d(J) : e(J) + \sum_{T \in \tau} rf^{(T)}(t) \leq t. \quad (1)$$

What about the Combinatorial Explosion?
Overapproximation: \textit{mrf}

- Approach: Define an overapproximation
- \( \text{mrf}^{(T)}(t) \): \textit{Maximum} of all \( \text{rf}^{(T)}(t) \) for a task \( T \)
  - “Request-Bound Function”
  - “Workload-Arrival Function”
- New test:
  \[
  \exists t \leq d(J) : e(J) + \sum_{T \in \tau} \text{mrf}^{(T)}(t) \leq t.
  \]
- \textit{Efficient}: Only \textit{one} test, no combinatorial explosion
Overapproximation: \textit{mrf}

- Approach: Define an overapproximation
- \( mrf^{(T)}(t): \textit{Maximum} \textit{ of all } rf^{(T)}(t) \textit{ for a task } T \)
  - “Request-Bound Function”
  - “Workload-Arrival Function”
- New test:
  \[ \exists t \leq d(J) : e(J) + \sum_{T \in \tau} mrf^{(T)}(t) \leq t. \]
- \textbf{Efficient}: Only \textit{one} test, no combinatorial explosion
- Problem: Imprecise!

---

\textbf{How can we get efficiency and precision?}
Overapproximation: \textit{mrf}

- Approach: Define an overapproximation
- \(\text{mrf}^{(T)}(t)\): \textit{Maximum} of all \(\text{rf}^{(T)}(t)\) for a task \(T\)
  - “Request-Bound Function”
  - “Workload-Arrival Function”
- New test:
  \[
  \exists t \leq d(J) : e(J) + \sum_{T \in \tau} \text{mrf}^{(T)}(t) \leq t.
  \]
- \textit{Efficient}: Only \textit{one} test, no combinatorial explosion
- Problem: Imprecise!

\[
\begin{align*}
J_1 &: \langle 2, 5 \rangle \rightarrow 20 \\
J_2 &: \langle 6, 30 \rangle \rightarrow 50
\end{align*}
\]

How can we get efficiency \textit{and} precision?
Abstraction Tree

Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is *mrf*, maximum of *all rf*
Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is *mrf*, maximum of *all rf*
Define an *abstraction tree* per task:

- Leaves are concrete $rf$
- Each node: maximum function of child nodes
- Root is $mrf$, maximum of all $rf$
Define an *abstraction tree* per task:

- Leaves are concrete $rf$
- Each node: maximum function of child nodes
- Root is $mrf$, maximum of *all* $rf$
Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is *mrf*, maximum of *all* *rf*
Combinatorial Abstraction Refinement

New Algorithm:

- Test *one* combination of all $mrf$.
- If schedulable: done
- Otherwise: Replace *one* $mrf$ with all child nodes,
  - 2 new combinations to test
- Repeat until:
  - All combinations show schedulability, or
  - A combination of leaves shows non-schedulability
Combinatorial Abstraction Refinement: Example

Task $T_1$

Task $T_3$

Task $T_4$

Testing $rf$ tuples:

$(A, A, A, A)$
Combinatorial Abstraction Refinement: Example

Testing \( rf \) tuples:

\[ (A, A, A, A, A) \]

Test: \( \exists t \leq d(J) : e(J) + \sum_{T \in \tau} rf(T)(t) \leq t \)
Combinatorial Abstraction Refinement: Example

Task $T_1$

Task $T_3$

Task $T_3$

Task $T_4$

Testing $rf$ tuples:

$\begin{pmatrix} A, A, A, A \end{pmatrix}$ UNSCHED
Combinatorial Abstraction Refinement: Example

Testing \( rf \) tuples:

\[(A, A, A, A)\]
\[(B, A, A, A)\]
\[(C, A, A, A)\]

Result: Schedulable!

Total combinations: \(3 \cdot 2 \cdot 4 \cdot 3 = 72\); Tested: 5 (!)
Combinatorial Abstraction Refinement: Example

Testing rf tuples:

\[
\begin{align*}
(A, A, A, A) & \quad \text{UNSCED} \\
(B, A, A, A) & \quad ? \\
(C, A, A, A) & \\
\end{align*}
\]

Test: \( \exists t \leq d(J) : e(J) + \sum_{T \in \tau} rf(T)(t) \leq t \)
Combinatorial Abstraction Refinement: Example

Task $T_1$

Task $T_3$

Task $T_3$

Task $T_4$

Testing $rf$ tuples:

- $(A, A, A, A, A)$: UNSCHED
- $(B, A, A, A, A)$: SCHED
- $(C, A, A, A, A)$

Result: Schedulable!

Total combinations: $3 \cdot 2 \cdot 4 \cdot 3 = 72$; Tested: 5 (!)
Combinatorial Abstraction Refinement: Example

Testing $rf$ tuples:

- $(A, A, A, A, A)$: UNSCHED
- $(B, A, A, A, A)$: SCHED
- $(C, A, A, A, A)$: ?

Test: $\exists t \leq d(J) : e(J) + \sum_{T \in \tau} rf(T)(t) \leq t$
Combinatorial Abstraction Refinement: Example

Task $T_1$

Task $T_3$

Task $T_3$

Task $T_4$

Testing $rf$ tuples:

$$(A, A, A, A, A) \quad \text{UNSCHEDED}$$

$$(B, A, A, A, A) \quad \text{SCHED}$$

$$(C, A, A, A, A) \quad \text{UNSCHEDED}$$

$$(C, A, A, A, A) \quad \text{UNSCHEDED}$$
Testing rf tuples:

- \((A, A, A, A)\) UNSCHED
- \((B, A, A, A)\) SCHED
- \((C, A, A, A)\) UNSCHED

Total combinations: \(3 \cdot 2 \cdot 4 \cdot 3 = 72\); Tested: 5 (!)
Combinatorial Abstraction Refinement: Example

Testing rf tuples:

- (A, A, A, A) \text{UNSCHED}
- (B, A, A, A) \text{SCHED}
- (C, A, A, A) \text{UNSCHED}
- (C, A, B, A) ?
- (C, A, C, A)

Test: $\exists t \leq d(J) : e(J) + \sum_{T \in \tau} rf(T)(t) \leq t$
Combinatorial Abstraction Refinement: Example

Testing \( rf \) tuples:

\[
\begin{align*}
(A, A, A, A) & \quad \text{UNSCHED} \\
(B, A, A, A) & \quad \text{SCHED} \\
(C, A, A, A) & \quad \text{UNSCHED} \\
(C, A, B, A) & \quad \text{SCHED} \\
(C, A, C, A) & \\
\end{align*}
\]
Combinatorial Abstraction Refinement: Example

Testing rf tuples:

- \((A, A, A, A)\) UNSCHED
- \((B, A, A, A)\) SCHED
- \((C, A, A, A)\) UNSCHED
- \((C, A, B, A)\) SCHED
- \((C, A, C, A)\) ?

Test: \(\exists t \leq d(J) : e(J) + \sum_{T \in \tau} rf(T)(t) \leq t\)
Combinatorial Abstraction Refinement: Example

Task $T_1$

Task $T_3$

Task $T_3$

Task $T_4$

Testing rf tuples:

- $(A, A, A, A)$: UNSCHED
- $(B, A, A, A)$: SCHED
- $(C, A, A, A)$: UNSCHED
- $(C, A, B, A)$: SCHED
- $(C, A, C, A)$: SCHED

Result: *Schedulable!*
Combinatorial Abstraction Refinement: Example

Testing rf tuples:

\[ \begin{align*}
&\text{(A, A, A, A)} & \text{UNSCHED} \\
&\text{(B, A, A, A)} & \text{SCHED} \\
&\text{(C, A, A, A)} & \text{UNSCHED} \\
&\text{(C, A, B, A)} & \text{SCHED} \\
&\text{(C, A, C, A)} & \text{SCHED}
\end{align*} \]

Result: \textit{Schedulable!}

Total combinations: \(3 \cdot 2 \cdot 4 \cdot 3 = 72\); Tested: 5 (!)
Problem Introduction
- Digraph Real-Time Tasks
- Complexity Results

Analysis Approach
- Request Functions
- Rf-based Test

Combinatorial Abstraction Refinement
- Abstraction Trees
- Refinement Procedure

Evaluation
1 Problem Introduction
   - Digraph Real-Time Tasks
   - Complexity Results

2 Analysis Approach
   - Request Functions
   - Rf-based Test

3 Combinatorial Abstraction Refinement
   - Abstraction Trees
   - Refinement Procedure

4 Evaluation
Comparing runtimes of
- EDF-test using dbf (pseudo-polynomial)
- SP-test based on Combinatorial Abstraction Refinement
Evaluation: Tested vs. Total Combinations

$10^5$ samples of single-job tests.

- Executed tests: in 99.9% of all cases, less than 100
- Total combinations possible: $10^{12}$ or more
Summary and Outlook

- Solve coNP-hard problem
  - Previously unsolved
  - Efficient method
- Abstraction refinement
  - General method
  - Combinatorial problems
  - Needs abstraction lattice

- Ongoing work:
  - Response-Time Analysis (submitted)
  - Apply to other problems
Q & A

Thanks!