On the Tractability of Digraph-Based Task Models

Martin Stigge

Uppsala University, Sweden

Joint work with Pontus Ekberg, Nan Guan and Wang Yi
Analysis of Abstract Models

- Model hard real-time systems
  - Analysis: Guarantee deadlines
  - *Expressiveness* of models?
  - *Efficiency* of analysis?

**Question**
- How expressive can a model be?
- ... with a *tractable* feasibility test?
Context: Real-Time Task Models

- System is composed of tasks, releasing jobs $J = (r, e, d)$
  - Release time $r$
  - Worst-case execution time $e$
  - Deadline $d$

Feasibility: *Can we schedule s.t. all jobs meet their deadlines?*

In this work:
- Preemptive schedules
- On uniprocessors
- Independent jobs

In this setting: EDF is optimal.

$Feasible \iff Schedulable$ with $EDF$
System is composed of **tasks**, releasing **jobs** $J = (r, e, d)$
- Release time $r$
- Worst-case execution time $e$
- Deadline $d$

**Feasibility:** *Can we schedule s.t. all jobs meet their deadlines?*

**In this work:**
- Preemptive schedules
- On uniprocessors
- Independent jobs

In this setting: EDF is optimal.

**Feasible $\iff$ Schedulable with EDF**
The Liu and Layland (L&L) Task Model
(Liu and Layland, 1973)

- Tasks are periodic
  - Job WCET $e$
  - Minimum inter-release delay $p$ (implicit deadline)

- Advantages: Well-known model; efficient schedulability test
- Disadvantage: Very limited expressiveness
Hierarchy of Models

- **L&L**
- **GMF**
- **RRT**
- **DRT**

- Periodic
- Different job types
- Branching, loops, ...

- Strongly (co)NP-hard
- Pseudo-Polynomial

Feasibility test:
- **difficult**
- **efficient**

Expressiveness:
- **high**
- **low**

Graphs:
- **arbitrary graph**
- **DAG**
- **cycle graph**
- **two integers**

- **DRT**
- **RRT**
- **GMF**
- **L&L**

- Expressiveness:
  - Pseudo-Polynomial
  - Strongly (co)NP-hard

- Feasibility test:
  - Efficient
  - Difficult
Hierarchy of Models

- **L&L**: two integers, periodic
- **GMF**: cycle graph, different job types
- **RRT**: DAG, branching
- **DRT**: arbitrary graph, branching, loops, ...

- **Feasibility test**
  - difficult
  - efficient

- **Expressiveness**
  - low
  - high

- **Complexity**
  - Strongly (co)NP-hard
  - Pseudo-Polynomial

Referenced works:
- [Liu et al., 1973]
- [Mok et al., 1999]
- [Baruah, 2003]
- [S. et al., 2011]
Hierarchy of Models

Feasibility test: efficient → difficult
Expressiveness: low → high

- L&L
- GMF
- RRT
- DRT

- period
- different job types
- branching, loops, ...

- Strongly (co)NP-hard
- Pseudo-Polynomial

- arbitrary graph
- DAG
- cycle graph
- two integers

- Feasibility test: efficient → difficult
- Expressiveness: low → high

Question: How close to this border?

k-EDRT

Feasibility test: efficient → difficult
Expressiveness: low → high

- Martin Stigge
- Tractability of Digraph-Based Models
Hierarchy of Models

Question: How close to this border?

Feasibility test

difficult

efficient

Expressiveness

high

low

- **L&L**
- **GMF**
- **RRT**
- **DRT**

- **EDRT**
- **k-EDRT**

- **DAG**
- **arbitrary graph**

- **RRT**
- **branching**

- **GMF**
- **different job types**

- **GMF**
- **periodic**

- **L&L**
- **two integers**

- **[Liu et al., 1973]**
- **[Mok et al., 1999]**
- **[Baruah, 2003]**
- **[S. et al., 2011]**

- Strongly NP-hard
- Pseudo-Polynomial

- low

- high

- efficient

- difficult

- **Expressiveness**

- **Feasibility test**

- **Expressiveness**

- **Feasibility test**
The Digraph Real-Time (DRT) Task Model

- Branching, cycles (loops), ...
- Directed graph for each task
  - Vertices $J$: jobs to be released (with WCET and deadline)
  - Edges $(J_i, J_j)$: minimum inter-release delays $p(J_i, J_j)$

Theorem (S. et al., RTAS 2011)

For DRT task systems $\tau$ with a utilization bounded by any $c < 1$, feasibility can be decided in pseudo-polynomial time.
The Digraph Real-Time (DRT) Task Model

- Branching, cycles (loops), ...
- Directed graph for each task
  - Vertices $J$: jobs to be released (with WCET and deadline)
  - Edges $(J_i, J_j)$: minimum inter-release delays $p(J_i, J_j)$

Theorem (S. et al., RTAS 2011)

For DRT task systems $\tau$ with a utilization bounded by any $c < 1$, feasibility can be decided in pseudo-polynomial time.
Hierarchy of Models

- **L&L**
- **GMF**
- **RRT**
- **DRT**

**Feasibility test**
- **difficult**
- **efficient**

**Expressiveness**
- **low**
- **high**

- **EDRT**
  - Strongly non-P-hard
  - Pseudo-polynomial
  - **k-EDRT**

**Feasibility test**
- **arbitrary graph**
- **DAG**
- **cycle graph**
- **two integers**

**Expressiveness**
- **high**
- **low**

**expressiveness**
- **branching, loops, ...**
  - [S. et al., 2011]
- **branching**
  - [Baruah, 2003]
- **different job types**
  - [Mok et al., 1999]
- **periodic**
  - [Liu et al., 1973]
Hierarchy of Models

- **L&L**
- **GMF**
- **RRT**
- **DRT**

- **periodic**
- **different job types**
- **branching, loops, ...**

- **Two integers**
- **Cycle graph**
- **DAG**

- **Strongly (co)NP-hard**
- **Pseudo-polynomial**

- **Efficient**
- **Difficult**

- **Expressiveness**
- **Feasibility test**
Extending DRT: Global Timing Constraints – This Work

- Delays in DRT only for adjacent jobs
- What about adding global delay constraints?

Motivation:
- Mode sub-structures
- Burstiness
- ...
Extended DRT (EDRT) – This Work

- Extends DRT with **global delay constraints**
- **Directed graph** for each task
  - Vertices $J$: jobs to be released (with WCET and deadline)
  - Edges $(J_i, J_j)$: minimum inter-release delays $p(J_i, J_j)$
  - $k$ global constraints $(J_i, J_j, \gamma)

Theorem (Our technical result)

- For $k$-EDRT task systems with bounded utilization, feasibility is
  1. **decidable in pseudo-polynomial time** if $k$ is constant, and
  2. **strongly coNP-hard** in general.
Extended DRT (EDRT) – This Work

- Extends DRT with *global delay constraints*
- *Directed graph* for each task
  - Vertices \( J \): jobs to be released (with WCET and deadline)
  - Edges \((J_i, J_j)\): minimum inter-release delays \( p(J_i, J_j) \)
  - \( k \) global constraints \((J_i, J_j, \gamma)\)

![Directed Graph](image)

**Theorem (Our technical result)**

*For \( k \)-EDRT task systems with bounded utilization, feasibility is*

1. *decidable in pseudo-polynomial time* if \( k \) is constant, and
2. *strongly coNP-hard* in general.
Hierarchy of Models

Feasibility test

difficult

Expressiveness

low

high

efficient

DRT

branching, loops, …
[S. et al., 2011]

RRT

branching
[Baruah, 2003]

G MF

different job types
[Mok et al., 1999]

L&L

periodic
[Liu et al., 1973]

arbitrary graph

DAG

cycle graph

two integers

EDRT

Strongly (co)NP-hard

k-EDRT

Pseudo-Polynomial
Fahrplan
Hardness for EDRT

- Number of constraints now \textit{not constant}
- Reduction from \textit{Hamiltonian Path Problem} (strongly NP-hard)
Hardness for EDRT

- Number of constraints now *not constant*
- Reduction from *Hamiltonian Path Problem* (strongly NP-hard)
Hardness for EDRT

- Number of constraints now *not constant*
- Reduction from *Hamiltonian Path Problem* (strongly NP-hard)
Analysing $k$-EDRT: The Problem

- Detour: Analyzing DRT
  - Using demand bound functions
  - Compute exec. demand and deadline for all paths in $G(T)$

Problem: Constraints ignored during path exploration
- $\langle 2, 4 \rangle$ is lacking constraint information
- ... about the active constraint
Analysing $k$-EDRT: The Problem

- Detour: Analyzing DRT
  - Using demand bound functions
  - Compute exec. demand and deadline for *all paths* in $G(T)$

```
 ⟨2,2⟩  J₂  2  J₃
 ↓   5   ↓   3
 J₁   J₄  7   J₅  2
 ⟨1,2⟩  ⟨1,2⟩  ⟨1,2⟩
 Demand pair: ⟨2,4⟩
```

- Problem: *Constraints* ignored during path exploration
  - ⟨2,4⟩ is lacking constraint information
  - ... about the *active constraint*
Analysing $k$-EDRT: The Problem

- **Detour: Analyzing DRT**
  - Using demand bound functions
  - Compute exec. demand and deadline for *all paths* in $G(T)$

![Diagram of a directed graph with nodes labeled $J_1, J_2, J_3, J_4, J_5$ and edges showing demand pairs.]

- **Problem:** *Constraints* ignored during path exploration
  - $\langle 2, 4 \rangle$ is lacking constraint information
  - ... about the *active constraint*
Analysing $k$-EDRT: The Problem

- **Detour: Analyzing DRT**
  - Using demand bound functions
  - Compute exec. demand and deadline for *all paths* in $G(T)$

```
J_1  J_2  J_3  J_4  J_5
\langle 1, 2 \rangle \quad \langle 1, 2 \rangle \quad \langle 2, 2 \rangle \quad \langle 1, 2 \rangle \quad \langle 4, 7 \rangle
\langle 2, 4 \rangle \quad \langle 4, 7 \rangle \quad \langle 4, 8 \rangle
\langle 2, 4 \rangle \quad \langle 3, 4 \rangle \quad \langle 2, 9 \rangle
```

**Demand pair:** $\langle 2, 4 \rangle$

**New demand pair:** $\langle 4, 7 \rangle$

- **Problem:** Constraints ignored during path exploration
  - $\langle 2, 4 \rangle$ is lacking constraint information
  - ... about the active constraint
Analysing $k$-EDRT: The Problem

- Detour: Analyzing DRT
  - Using demand bound functions
  - Compute exec. demand and deadline for *all paths* in $G(T)$

- Problem: *Constraints* ignored during path exploration
  - $\langle 2, 4 \rangle$ is lacking constraint information
  - ... about the *active constraint*
Analysing $k$-EDRT: Solution

- Translate $k$-EDRT into *plain DRT*
  - Represent active constraints as *countdowns*
  - Store countdown values in DRT vertices
  - Preserve demand bound function

Optimizations: Efficient on-the-fly translation
Analysing $k$-EDRT: Solution

- Translate $k$-EDRT into *plain DRT*
  - Represent active constraints as *countdowns*
  - Store countdown values in DRT vertices
  - Preserve demand bound function

Optimizations: Efficient on-the-fly translation
Analysing $k$-EDRT: Solution

- Translate $k$-EDRT into *plain DRT*
  - Represent active constraints as *countdowns*
  - Store countdown values in DRT vertices
  - Preserve demand bound function

\[
\begin{align*}
J_1 & \rightarrow J_2, 5 \\
J_2 & \rightarrow J_3, k \\
J_3 & \rightarrow J_1, 5 \\
J_1, 5 & \rightarrow J_2, 3 \\
J_2, 3 & \rightarrow J_3, 0 \\
J_3, 0 & \rightarrow J_2, 0
\end{align*}
\]

Optimizations: Efficient on-the-fly translation
Analysing \( k \)-EDRT: Solution

- Translate \( k \)-EDRT into plain DRT
  - Represent active constraints as countdowns
  - Store countdown values in DRT vertices
  - Preserve demand bound function

\[
\begin{align*}
J_1 & \rightarrow J_2 & J_1 & \rightarrow J_2 \\
2 & \rightarrow 5 & 2 & \rightarrow 3 \\
J_2 & \rightarrow J_3 & J_2 & \rightarrow J_3 \\
2 & \rightarrow 0 & 3 & \rightarrow 2 \\
J_3 & & & \\
\end{align*}
\]

\( 1 \)-EDRT task

\[
\begin{align*}
J_1 & \rightarrow J_2 & J_1 & \rightarrow J_2 \\
5 & \rightarrow 0 & 5 & \rightarrow 2 \\
J_2 & \rightarrow J_3 & J_2 & \rightarrow J_3 \\
0 & \rightarrow 0 & 0 & \rightarrow 0 \\
J_3 & & & \\
\end{align*}
\]

Translated DRT task

Optimizations: Efficient on-the-fly translation
Analysing \( k \)-EDRT: Solution

- Translate \( k \)-EDRT into *plain DRT*
  - Represent active constraints as *countdowns*
  - Store countdown values in DRT vertices
  - Preserve demand bound function

**1-EDRT task**

**Translated DRT task**

- Optimizations: Efficient on-the-fly translation
Summary and Outlook

- Introduced Extension to DRT Task Model
  - Global delay constraints
- Establishes tractability borderline for feasibility test
  - Constant number of constraints: tractable
  - Unbounded number of constraints: intractable

- Ongoing work:
  - Global constraints for simpler models (RRT, GMF)
  - Interaction with Resource Sharing Protocols (cf. talk tomorrow)
Q & A

Thanks!