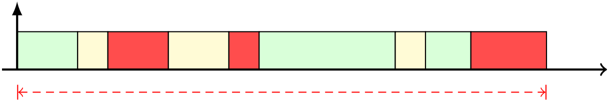
Refinement-based Exact Response-Time Analysis

Martin Stigge

Uppsala University, Sweden

Joint work with Nan Guan and Wang Yi

Response-Time Analysis



- Useful for
  - Schedulability analysis
  - Jitters in larger systems
  - ...
- Standard RTA for static priorities + periodic/sporadic tasks

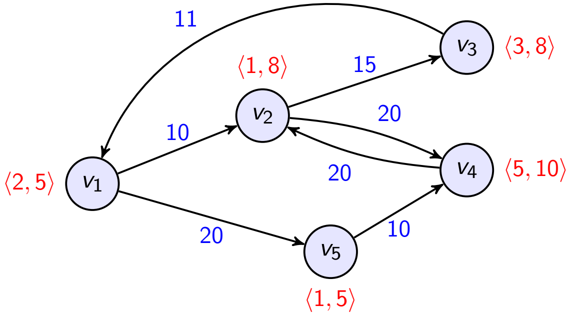
$$R_j = C_j + \sum_{i \in hp(j)} \left\lceil \frac{R_j}{T_i} \right\rceil C_i$$

Not everything is periodic!

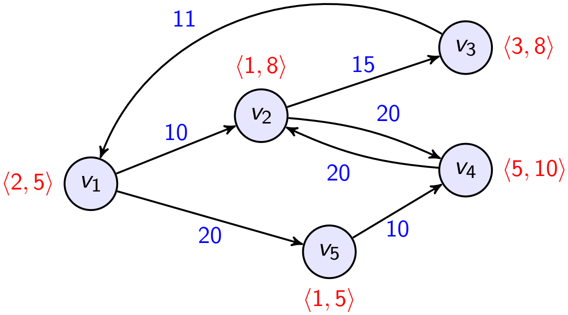
The Digraph Real-Time (DRT) Task Model

(S. et al., RTAS 2011)

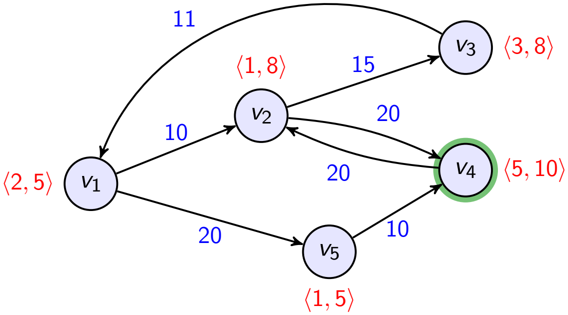
- Generalizes periodic, sporadic, GMF, RRT, ...
- *Directed graph* for each task
  - Vertices v: jobs to be released (with WCET and deadline)
  - Edges (u, v): minimum inter-release delays $p(u, v)$



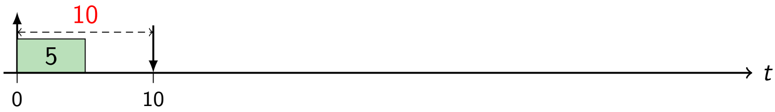
DRT: Semantics



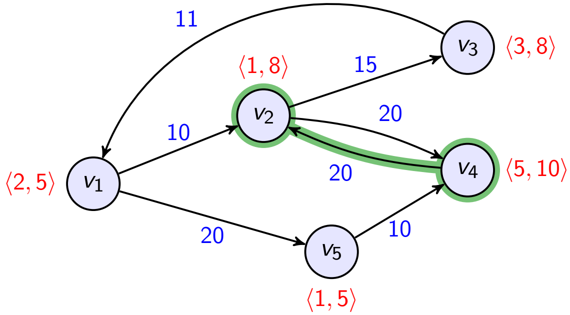
DRT: Semantics



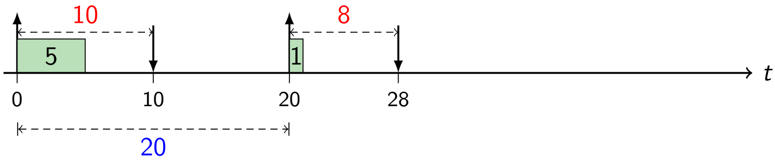
Path $\pi = (v_4)$



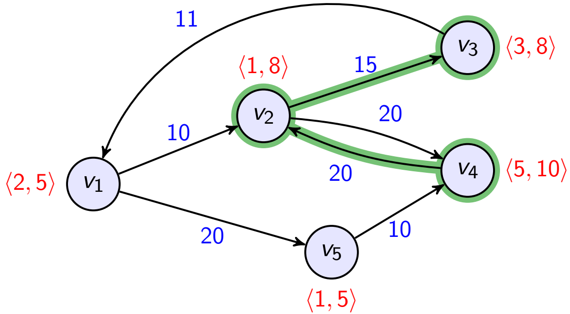
DRT: Semantics



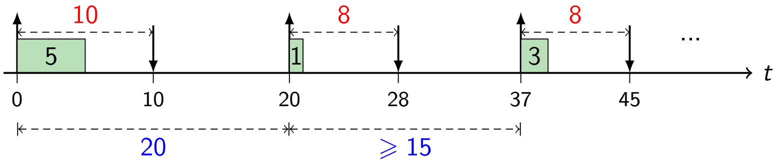
Path $\pi = (v_4, v_2)$



DRT: Semantics

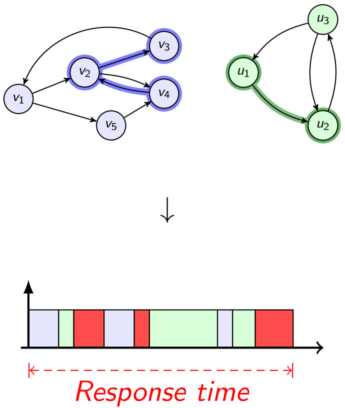


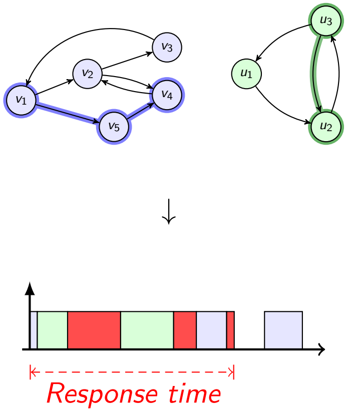
Path $\pi = (v_4, v_2, v_3)$



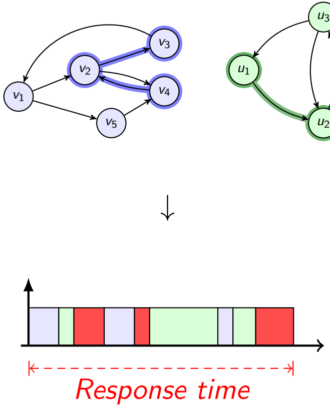
Response-Time Analysis for DRT

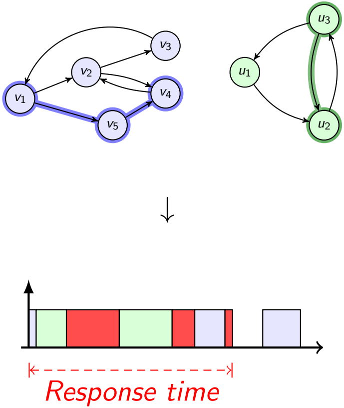
Problem: Path Combinations





Problem: Path Combinations





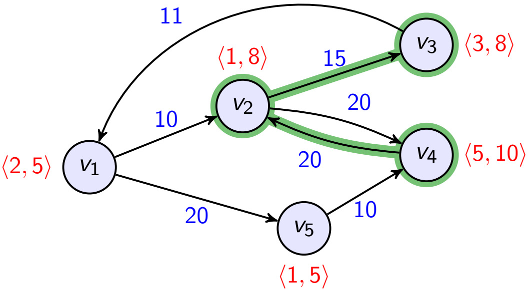
Combinatorial Explosion!

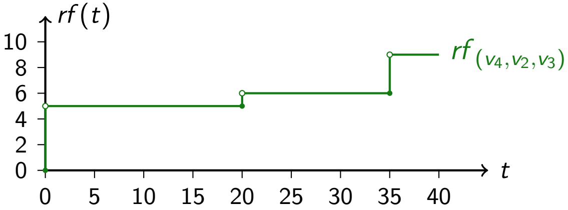
Fahrplan

Fahrplan

Step 1: From Paths to Functions

Step 1: From Paths to Functions





Request Functions

Useful for deriving response time:

$$R_{SP}(v,\bar{rf}) = \min\left\{ t \geqslant 0 \mid e(v) + \sum_{T' > T} rf^{(T')}(t) \leqslant t \right\}$$

$$R_{SP}(v) = \max_{\bar{rf} \in RF(\tau)} R_{SP}(v,\bar{rf})$$

Request Functions

Useful for deriving response time:

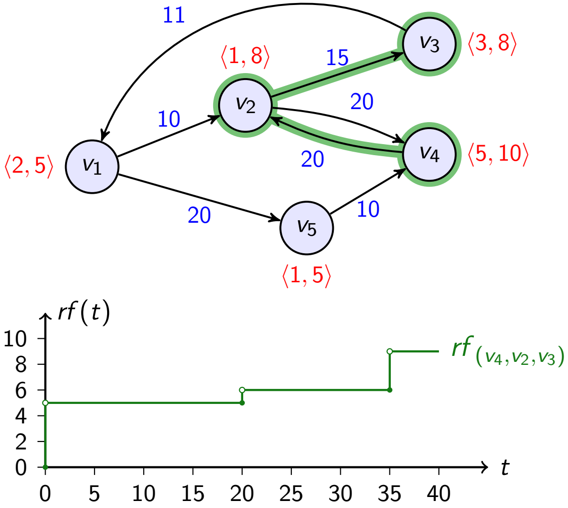
$$R_{SP}(v, \bar{rf}) = \min \left\{ t \geqslant 0 \mid e(v) + \sum_{T' > T} rf^{(T')}(t) \leqslant t \right\}$$

$$R_{SP}(v) = \max_{\bar{rf} \in RF(\tau)} R_{SP}(v, \bar{rf})$$

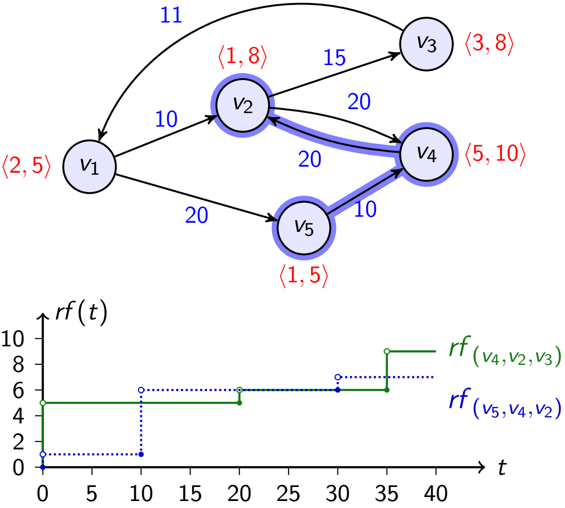
Combinatorial Explosion?!

Step 2: Abstraction Trees

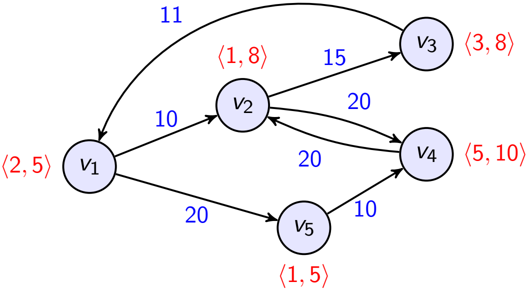
Abstract Request Functions

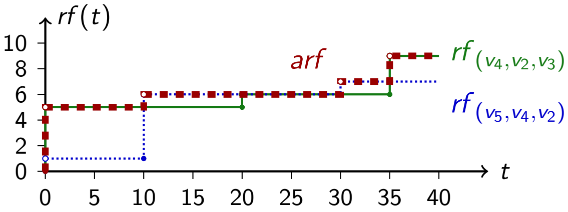


Abstract Request Functions

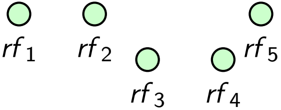


Abstract Request Functions





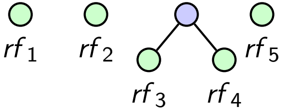
Abstraction Tree



Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all* *rf*

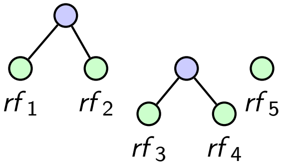
Abstraction Tree



Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all* *rf*

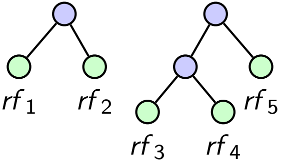
Abstraction Tree



Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all* *rf*

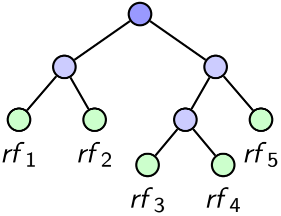
Abstraction Tree



Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all* *rf*

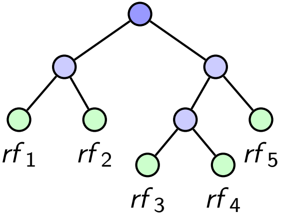
Abstraction Tree



Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all* *rf*

Abstraction Tree



Define an *abstraction tree* per task:

- Leaves are concrete *rf*
- Each node: maximum function of child nodes
- Root is maximum of *all* *rf*

Allows stepwise refinement!

Step 3: Refinement Algorithm

Step 3: Refinement Algorithm

Store

Tuple: $\bar{rf} = (rf^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$

Step 3: Refinement Algorithm

Store

$(23, \bar{rf}_1)$

Tuple: $\bar{rf} = (rf^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$

$\downarrow$

Response time: $R_{SP}(v, \bar{rf}) = 23$

Using: $R_{SP}(v, \bar{rf}) = \min \left\{ t \geqslant 0 \mid e(v) + \sum_{T' > T} rf^{(T')}(t) \leqslant t \right\}$

Step 3: Refinement Algorithm

Store

$(23, \bar{rf}_1)$

Step 3: Refinement Algorithm

Step:

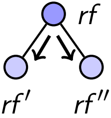
$\bar{rf}_1 = (rf^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$

$\downarrow$

$\bar{rf}_2 = (rf'^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$

$\bar{rf}_3 = (rf''^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$

In T_1:



Store

$(23, \bar{rf}_1)$

Step 3: Refinement Algorithm

Step:

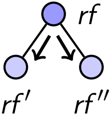
$$\bar{rf}_1 = (rf^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$$

$$\downarrow$$

$$\bar{rf}_2 = (rf'^{(T_1)}, rf^{(T_2)}, rf^{(T_3)}) \rightarrow 18$$

$$\bar{rf}_3 = (rf''^{(T_1)}, rf^{(T_2)}, rf^{(T_3)}) \rightarrow 21$$

In T_1:



Store

$(23, \bar{rf}_1)$

Step 3: Refinement Algorithm

Step:

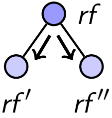
$$\bar{rf}_1 = (rf^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$$

$$\downarrow$$

$$\bar{rf}_2 = (rf'^{(T_1)}, rf^{(T_2)}, rf^{(T_3)}) \rightarrow 18$$

$$\bar{rf}_3 = (rf''^{(T_1)}, rf^{(T_2)}, rf^{(T_3)}) \rightarrow 21$$

In T_1:



Store

$\cancel{(23, \bar{rf}_1)}$

$(21, \bar{rf}_2)$

$(18, \bar{rf}_3)$

Step 3: Refinement Algorithm

Store

$(21, \bar{rf}_2)$

$(18, \bar{rf}_3)$

Step 3: Refinement Algorithm

Store

$(21, \bar{rf}_2)$

$(18, \bar{rf}_3)$

Step:

$\bar{rf}_2 = (rf^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$

Step 3: Refinement Algorithm

Step:

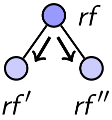
$$\bar{rf}_2 = (rf^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$$

$$\downarrow$$

$$\bar{rf}_4 = (rf^{(T_1)}, rf'^{(T_2)}, rf^{(T_3)})$$

$$\bar{rf}_5 = (rf^{(T_1)}, rf''^{(T_2)}, rf^{(T_3)})$$

In T_2:



Store

$(21, \bar{rf}_2)$

$(18, \bar{rf}_3)$

Step 3: Refinement Algorithm

Step:

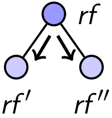
$$\bar{rf}_2 = (rf^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$$

$$\downarrow$$

$$\bar{rf}_4 = (rf^{(T_1)}, rf'^{(T_2)}, rf^{(T_3)}) \rightarrow 20$$

$$\bar{rf}_5 = (rf^{(T_1)}, rf''^{(T_2)}, rf^{(T_3)}) \rightarrow 17$$

In T_2:



Store

$(21, \bar{rf}_2)$

$(18, \bar{rf}_3)$

Step 3: Refinement Algorithm

Step:

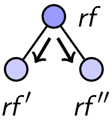
$$\bar{rf}_2 = (rf^{(T_1)}, rf^{(T_2)}, rf^{(T_3)})$$

$$\downarrow$$

$$\bar{rf}_4 = (rf^{(T_1)}, rf'^{(T_2)}, rf^{(T_3)}) \rightarrow 20$$

$$\bar{rf}_5 = (rf^{(T_1)}, rf''^{(T_2)}, rf^{(T_3)}) \rightarrow 17$$

In T_2:



Store

$\cancel{(21, \bar{rf}_2)}$

$(20, \bar{rf}_4)$

$(18, \bar{rf}_3)$

$(17, \bar{rf}_5)$

Step 3: Refinement Algorithm

Store

$(20, \overline{rf}_4)$

$(18, \overline{rf}_3)$

$(17, \overline{rf}_5)$

$\vdots$

Step 3: Refinement Algorithm

Initialization:

- Most abstract functions

Each iteration:

- Replace functions along *abstraction trees*

Termination:

- All functions are *concrete*

Store

$(20, \bar{rf}_4)$

$(18, \bar{rf}_3)$

$(17, \bar{rf}_5)$

$\vdots$

Step 3: Refinement Algorithm

Initialization:

- Most abstract functions

Each iteration:

- Replace functions along *abstraction trees*

Termination:

- All functions are *concrete*

Store

$(20, \bar{rf}_4)$

$(18, \bar{rf}_3)$

$(17, \bar{rf}_5)$

$\vdots$

Step 3: Refinement Algorithm

Initialization:

- Most abstract functions

Each iteration:

- Replace functions along *abstraction trees*

Termination:

- All functions are *concrete*

Store

$(20, \overline{rf}_4)$

$(18, \overline{rf}_3)$

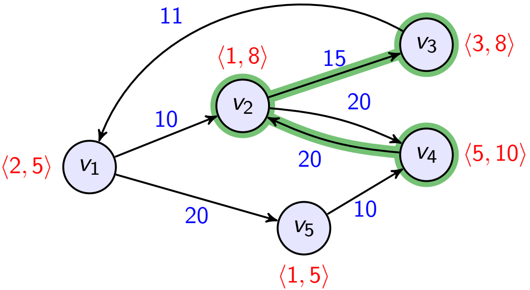
$(17, \overline{rf}_5)$

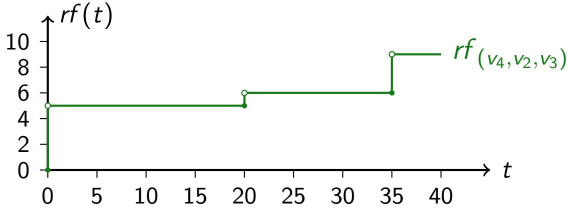
$\vdots$

Pluggable Path Abstractions!

Path Abstractions: SP + EDF

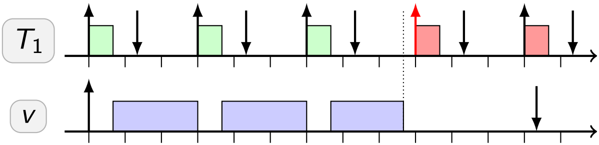
Path Abstractions: Static Priorities

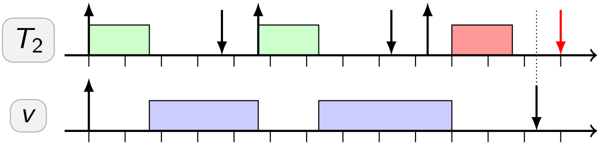




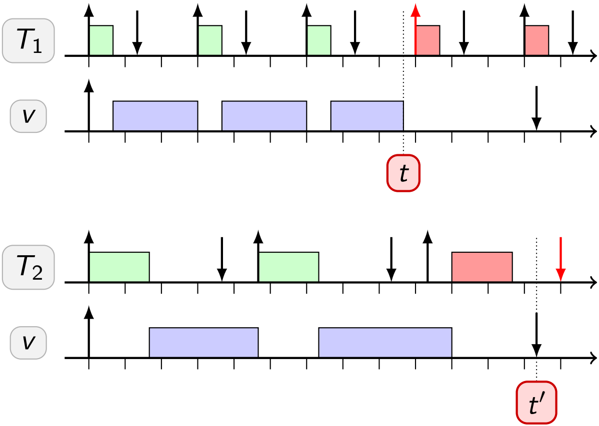
$$rf_\pi(t) := \max \left\{ e(\pi') \mid \pi' \text{ is prefix of } \pi \text{ and } p(\pi') < t \right\}$$

Path Abstractions: EDF





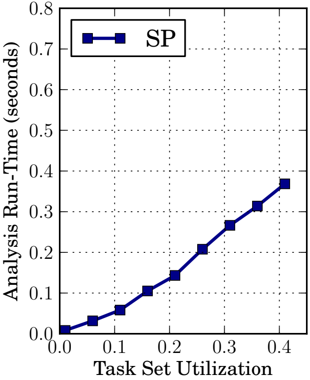
Path Abstractions: EDF

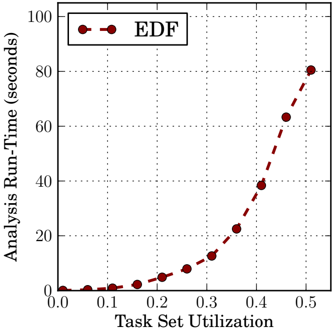


$$wf_\pi(t,t') := \max\{e(\pi') \mid \pi' \text{ is prefix of } \pi, \; p(\pi') < t \text{ and } d(\pi') \leqslant t'\}.$$

Evaluation

Evaluation: Run-time Scaling

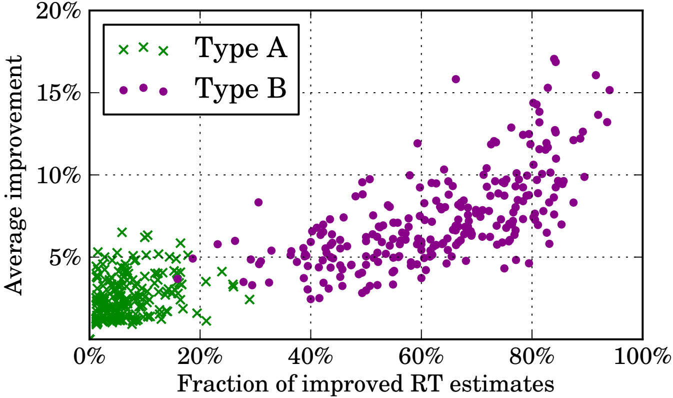




10-20 tasks with 5-10 vertices each, branching degree 1-3

(Busy window extension for EDF.)

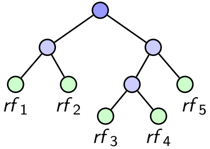
Evaluation: Precision Improvement



Type A: lower parameter variance
Type B: higher parameter variance

Summary

- Exact solution for NP-hard problem
  - *Efficient* method
  - Iterative refinement
- Pluggable path abstractions
  - Static Priorities
  - EDF
  - *Flexible*
- Ongoing work:
  - Apply to other problems



Q & A

Thanks!