Optimal & Real Time Scheduling using Model Checking Technology
AMETIST
advanced methods for timed systems

April 2002 – June 2005 IST-2001-35304

- Academic partners:
  - Nijmegen
  - Aalborg
  - Dortmund
  - Grenoble
  - Marseille
  - Twente
  - Weizmann

- Industrial Partners:
  - Axxom
  - Bosch
  - Cybernetix
  - Terma
AMETIST
advanced methods for timed systems

OBJECTIVES

- powerful, unifying mathematical modelling
- efficient computerized problem-solving tools
- distributed real-time systems
- time-dependent behaviour and dynamic resource allocation
- TIMED AUTOMATA

SIDMAR Overview

GOAL: Maximize utilization of plant

INPUT sequence of steel loads ("pigs")

OUTPUT sequence of higher quality steel

SIDMAR Modelling

A Single Load

UPPAAL Model
CPS: Informal description

- CPS obtains and makes available for other systems information about environment of a car. This information may be used for:
  - Parking assistance
  - Pre-crash detection
  - Blind spot supervision
  - Lane change supervision
  - Stop & go
  - Etc

The CPS considered in this case study

- One sensor group only (currently 2 sensors)
- Only the front sensors and corresponding controllers
- Application: pre-crash detection, parking assistance, stop & go

CPS:
- Parking assistance
- Pre-crash detection
- Blind spot supervision
- Lane change supervision
- Stop & go
- Etc

Case Studies

- Cybernetix: Smart Card Personalization
- Terma: Memory Interface
- Bosch: Car Periphery Sensing
- AXXOM: Lacquer Production
- Benchmarks

Product flow of a Product

- Dispersion
- Dose Spinner
- Mixing Vessel
- Laboratory
- Filling Stations
- Storage

Classical Laboratory
Welcome!

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).

UPPAAL CORA is a branch of UPPAAL for Cost Optimal Reachability Analysis developed by the UPPAAL team as part of the VH3 and AMETIST projects. Whereas UPPAAL supports model checking of timed automata, UPPAAL CORA uses an extension of timed automata called LPTA. LPTA allows you to annotate the model with the notion of cost. This can be the cost of delay in certain situations or the cost of particular actions. UPPAAL CORA then finds optimal paths matching goal conditions.

UPPAAL CORA has been used in a number of case studies. Some of these are described on the case study page of this site. If you come up with interesting uses, please contact us. We are interested in hearing what you do!

Due to different internal data structures, UPPAAL CORA currently consists of two different versions:

- A version for the simplified case of time optimal reachability analysis.
- A version for the full language of LPTA.

Latest News

UPPAAL Got New Home Page
25 Jan 2005

The main UPPAAL site adopted the same layout as the UPPAAL CORA site. At the same time, the UPPAAL CORA site has adopted a new color scheme.

Updated case studies
30 Nov 2004

The models of the case studies can now be downloaded from the case study page.
Overview

- Timed Automata & Scheduling
- Priced Timed Automata and Optimal Scheduling
- Optimal Infinite Scheduling
- Optimal Scheduling Using Priced Timed Automata.
  G. Behrmann, K. G. Larsen, J. I. Rasmussen,
  ACM SIGMETRICS Performance Evaluation Review
- Optimal Scheduling and Off-Line Test Generation
Rush Hour

OBJECTIVE: Get your CAR out

Your CAR

EXIT

OBJ ECTI VE: Get your CAR out
Rush Hour

\[
\begin{align*}
\text{Red} & \quad \text{no} = 3, \quad i = 2 \\
& \quad \text{no} := no - 1, \quad \text{pos} := 2 \\
& \quad \text{pos} := \text{pos} - 1 \\
& \quad \text{left?} \\
& \quad \text{BOARD}[3][\text{pos} - 1] := 0 \\
& \quad \text{BOARD}[3][\text{pos} - 1] := 1 \\
& \quad \text{BOARD}[3][\text{pos} + 2 - 1] := 0 \\
& \quad \text{pos} := \text{pos} - 1 \\
& \quad \text{no} = 3, \quad i < 2 \\
& \quad \text{no} := no - 1, \quad \text{pos} := 2 \\
& \quad \text{pos} := \text{pos} + 1 \\
& \quad \text{right?} \\
& \quad \text{BOARD}[3][\text{pos} + 2] := 0 \\
& \quad \text{BOARD}[3][\text{pos} + 2] := 1 \\
& \quad \text{BOARD}[3][\text{pos}] := 0 \\
& \quad \text{pos} := \text{pos} + 1 \\
\end{align*}
\]
Rush Hour

![Diagram of Rush Hour game]
Real Time Scheduling

• Only 1 “Pass”
• Cheat is possible  
  (drive close to car with “Pass”)

SAFELY

CAN THEY MAKE IT TO SAFE  
WITHIN 70 MINUTES ???

UNSAFE

Crossing Times

5
10
20
25
Real Time Scheduling

Solve Scheduling Problem using UPPAAL
Steel Production Plant

- A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson
- Case study of Esprit-LTR project 26270 VHS
- Physical plant of SIDMAR located in Gent, Belgium.
- Part between blast furnace and hot rolling mill.

Objective: model the plant, obtain schedule and control program for plant.
**Steel Production Plant**

**Input:** sequence of steel loads ("pigs").

Load follows **Recipe** to obtain certain quality, e.g:

start; T1@10; T2@20; T3@10; T2@10; end within 120.

**Output:** sequence of higher quality steel.

∑=127
A single load (part of) Crane B
Controller Synthesis for LEGO Model

- LEGO RCX Mindstorms.
- Local controllers with control programs.
- IR protocol for remote invocation of programs.
- Central controller.

1971 lines of RCX code (n=5), 24860 - “ (n=60).
Timed Automata

Resource

Reset

Invariant

Synchronization

Guard

Semantics:

(Idle, x=0)

→ (Idle, x=2.5) d(2.5)
→ (InUse, x=0) use?
→ (InUse, x=5) d(5)
→ (Idle, x=5) done!
→ (Idle, x=8) d(3)
→ (InUse, x=0) use?

[Alur & Dill’89]
Composition

Resource

Idle

use?

x:=0

InUse

x<=B

x:=B

done!

Synchronization

(Idle, Init, B=0, x=0)

→ (Idle, Init, B=0, x=3.1415) d(3)

→ (InUse, Using, B=6, x=0) use

→ (InUse, Using, B=6, x=6) d(6)

→ (Idle, Done, B=6, x=6) done

Task

Init

use!

Using

B:=6

done?

Done

Shared variable
# Jobshop Scheduling

## Resources

<table>
<thead>
<tr>
<th></th>
<th>Sport</th>
<th>Economy</th>
<th>Local News</th>
<th>Comic Stip</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kim</strong></td>
<td>2. 5 min</td>
<td>4. 1 min</td>
<td>3. 3 min</td>
<td>1. 10 min</td>
</tr>
<tr>
<td><strong>Maria</strong></td>
<td>1. 10 min</td>
<td>2. 20 min</td>
<td>3. 1 min</td>
<td>4. 1 min</td>
</tr>
<tr>
<td><strong>Nicola</strong></td>
<td>4. 1 min</td>
<td>1. 13 min</td>
<td>3. 11 min</td>
<td>2. 11 min</td>
</tr>
</tbody>
</table>

**Problem:** compute the minimal **MAKESPAN**
Jobshop Scheduling in UPPAAL

<table>
<thead>
<tr>
<th></th>
<th>Sport</th>
<th>Economy</th>
<th>Local News</th>
<th>Comic Stip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kim</td>
<td>2.</td>
<td>5 min</td>
<td>4.</td>
<td>1 min</td>
</tr>
<tr>
<td>Maria</td>
<td>1.</td>
<td>10 min</td>
<td>2.</td>
<td>20 min</td>
</tr>
<tr>
<td>Nicola</td>
<td>4.</td>
<td>1 min</td>
<td>1.</td>
<td>13 min</td>
</tr>
</tbody>
</table>
## Experiments

**B-&-B algorithm running for 60 sec.**

### Lawrence Job Shop Problems

<table>
<thead>
<tr>
<th>Problem Instance</th>
<th>BF</th>
<th>MC</th>
<th>MC⁺</th>
<th>DF</th>
<th>RDF</th>
<th>Comb. Heur.</th>
<th>Minimal Makespan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cost</td>
<td>States</td>
<td>Cost</td>
<td>States</td>
<td>Cost</td>
<td>States</td>
<td>Cost</td>
</tr>
<tr>
<td>la01</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>la02</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>la03</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>la04</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>la05</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>593</td>
</tr>
<tr>
<td>la06</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3656</td>
</tr>
<tr>
<td>la07</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3410</td>
</tr>
<tr>
<td>la08</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3520</td>
</tr>
<tr>
<td>la09</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3984</td>
</tr>
<tr>
<td>la10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3681</td>
</tr>
<tr>
<td>la11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4974</td>
</tr>
<tr>
<td>la12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4557</td>
</tr>
<tr>
<td>la13</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4846</td>
</tr>
<tr>
<td>la14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1292</td>
</tr>
<tr>
<td>la15</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5264</td>
</tr>
<tr>
<td>la16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4849</td>
</tr>
<tr>
<td>la17</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4299</td>
</tr>
<tr>
<td>la18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4763</td>
</tr>
<tr>
<td>la19</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4566</td>
</tr>
<tr>
<td>la20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5056</td>
</tr>
<tr>
<td>la21</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7608</td>
</tr>
<tr>
<td>la22</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6920</td>
</tr>
<tr>
<td>la23</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7676</td>
</tr>
<tr>
<td>la24</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7237</td>
</tr>
<tr>
<td>la25</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7141</td>
</tr>
</tbody>
</table>
Task Graph Scheduling

Optimal Static Task Scheduling

- Task $P = \{P_1, \ldots, P_m\}$
- Machines $M = \{M_1, \ldots, M_n\}$
- Duration $\Delta : (P \times M) \to \mathbb{N}_\infty$
- $< :$ p.o. on $P$ (pred.)
Task Graph Scheduling

Optimal Static Task Scheduling

- Task $\mathbf{P} = \{P_1, \ldots, P_m\}$
- Machines $\mathbf{M} = \{M_1, \ldots, M_n\}$
- Duration $\Delta : (\mathbf{P} \times \mathbf{M}) \rightarrow \mathbb{N}_\infty$
- $\prec$ : p.o. on $\mathbf{P}$ (pred.)

- A task can be executed only if all predecessors have completed
- Each machine can process at most one task at a time
- Task cannot be preempted.

$\mathbf{M} = \{M_1, M_2\}$
Task Graph Scheduling

Optimal Static Task Scheduling

- Task \( P = \{P_1, \ldots, P_m\} \)
- Machines \( M = \{M_1, \ldots, M_n\} \)
- Duration \( \Delta : (P \times M) \rightarrow \mathbb{N}_{\infty} \)
- \(< : \text{p.o. on } P \text{ (pred.)} \)

\[ \begin{align*}
\text{Task} & \quad P_1 & \quad P_2 & \quad P_3 & \quad P_4 & \quad P_5 & \quad P_6 & \quad P_7 \\
\text{Duration} & \quad 16,10 & \quad 2,3 & \quad 6,6 & \quad 10,16 & \quad 2,2 & \quad 8,2 & \quad \\
M & \quad = \{M_1, M_2\} 
\end{align*} \]
# Experimental Results

<table>
<thead>
<tr>
<th>name</th>
<th>#tasks</th>
<th>#chains</th>
<th># machines</th>
<th>optimal</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>437</td>
<td>125</td>
<td>4</td>
<td>1178</td>
<td>1182</td>
</tr>
<tr>
<td>000</td>
<td>452</td>
<td>43</td>
<td>20</td>
<td>537</td>
<td>537</td>
</tr>
<tr>
<td>018</td>
<td>730</td>
<td>175</td>
<td>10</td>
<td>700</td>
<td>704</td>
</tr>
<tr>
<td>074</td>
<td>1007</td>
<td>66</td>
<td>12</td>
<td>891</td>
<td>894</td>
</tr>
<tr>
<td>021</td>
<td>1145</td>
<td>88</td>
<td>20</td>
<td>605</td>
<td>612</td>
</tr>
<tr>
<td>228</td>
<td>1187</td>
<td>293</td>
<td>8</td>
<td>1570</td>
<td>1574</td>
</tr>
<tr>
<td>071</td>
<td>1193</td>
<td>124</td>
<td>20</td>
<td>629</td>
<td>634</td>
</tr>
<tr>
<td>271</td>
<td>1348</td>
<td>127</td>
<td>12</td>
<td>1163</td>
<td>1164</td>
</tr>
<tr>
<td>237</td>
<td>1566</td>
<td>152</td>
<td>12</td>
<td>1340</td>
<td>1342</td>
</tr>
<tr>
<td>231</td>
<td>1664</td>
<td>101</td>
<td>16</td>
<td>t.o.</td>
<td>1137</td>
</tr>
<tr>
<td>235</td>
<td>1782</td>
<td>218</td>
<td>16</td>
<td>t.o.</td>
<td>1150</td>
</tr>
<tr>
<td>233</td>
<td>1980</td>
<td>207</td>
<td>19</td>
<td>1118</td>
<td>1121</td>
</tr>
<tr>
<td>294</td>
<td>2014</td>
<td>141</td>
<td>17</td>
<td>1257</td>
<td>1261</td>
</tr>
<tr>
<td>295</td>
<td>2168</td>
<td>965</td>
<td>18</td>
<td>1318</td>
<td>1322</td>
</tr>
<tr>
<td>292</td>
<td>2333</td>
<td>318</td>
<td>3</td>
<td>8009</td>
<td>8009</td>
</tr>
<tr>
<td>298</td>
<td>2399</td>
<td>303</td>
<td>10</td>
<td>2471</td>
<td>2473</td>
</tr>
</tbody>
</table>
Optimal Task Graph Scheduling

Power-Optimality

- Energy-rates: $C : M \rightarrow N \times N$

Diagram showing task graph with nodes and edges labeled with power consumption values.
Priced Timed Automata
Optimal Scheduling
Priced Timed Automata
Timed Automata + cost variable

$\begin{align*}
x &:= 0 \\
c' &:= 4 \\
\text{cost rate}
\end{align*}$

$\begin{align*}
x &:= 0 \\
0 &\leq y &\leq 4 \\
c &:= 1 \\
\text{cost rate}
\end{align*}$

$\begin{align*}
x &:= 0 \\
3 &\leq y \\
c &:= 4 \\
\text{cost update}
\end{align*}$

$\begin{align*}
x &:= 2 \\
2 &\leq y \\
c &:= 4 \\
\text{cost update}
\end{align*}$

$\begin{align*}
\text{Behrmann, Fehnker, et al. (HSCC'01)} \\
\text{Alur, Torre, Pappas (HSCC'01)}
\end{align*}$
Priced Timed Automata

Timed Automata + **COST** variable

\[
\begin{align*}
\text{(l}_1\text{,}x=y=0) & \xrightarrow{\varepsilon(3)}_{12} (\text{l}_1\text{,}x=y=3) \\
& \xrightarrow{1} (\text{l}_2\text{,}x=0,y=3) \\
& \xrightarrow{4} (\text{l}_3\text{,}_-,_-)
\end{align*}
\]

\[\sum c=17\]
Priced Timed Automata
Timed Automata + **COST** variable

---

**Problem:**
Find the **minimum** cost of reaching location $l_3$

---

Efficient Implementation:  
CAV’01 and TACAS’04

---

$\sum c=16$

$\sum c=11$

---

BRICS  
Basic Research  
in Computer Science
Aircraft Landing Problem

Planes have to keep separation distance to avoid turbulences caused by preceding planes.

- $E$: earliest landing time
- $T$: target time
- $L$: latest time
- $e$: cost rate for being early
- $l$: cost rate for being late
- $d$: fixed cost for being late

Cost function:

\[ \text{cost} = e^*(T-t) + d + l^*(t-T) \]
Planes have to keep separation distance to avoid turbulences caused by preceding planes.

Runway handles 2 types of planes:

129: Earliest landing time
153: Target landing time
559: Latest landing time
10: Cost rate for early
20: Cost rate for late

Modeling ALP with PTA
Symbolic "A*"
Zones

Definition
A zone $Z$ over a set of clocks $C$ is a finite conjunction of simple constraints of the forms:

$$\begin{align*}
x & \geq l \\
x & \leq u \\
x - y & \geq l' \\
x - y & \leq u'
\end{align*}$$

where $x, y \in C$, $l, u \in \mathbb{N}$ and $l', u' \in \mathbb{Z}$.

For $u \in \mathbb{R}^C$ and $Z$ a zone we write $u \models Z$ if $u$ satisfies all constraints of $Z$.

Operations

Reset: $\{x\}Z = \{u[0/x] | u \models Z\}$

Delay: $Z^{\uparrow} = \{u + d | u \models Z\}$

Offset: $\Delta_Z \models Z$ such that $\forall u \models Z. \forall x \in C. \Delta_Z(x) \leq u(x)$. 
Priced Zone

Definition
A priced zone $P$ is a tuple $(Z, c, r)$, where:

- $Z$ is a zone
- $c \in \mathbb{N}$ describes the cost of $\Delta_Z$
- $r : C \to \mathbb{Z}$ gives a rate for any clock $x \in C$.

We write $u \models P$ whenever $u \models Z$. For $u \models P$ we define $\text{Cost}(u, P)$ as follows:

$$\text{Cost}(u, P) = c + \sum_{x \in C} r(x) \cdot (u(x) - \Delta_Z(x))$$

Cost$(x, y) = 2y - x + 2$
Branch & Bound Algorithm

Cost := \infty
Passed := \emptyset
Waiting := \{(l_0, Z_0)\}

while Waiting \neq \emptyset do
    select \((l, Z)\) from Waiting
    if \(l = l_g\) and \(\text{minCost}(Z) < \text{Cost}\) then
        Cost := \text{minCost}(Z)
    end if
    if \(\text{minCost}(Z) + \text{Rem}(l, Z) \geq \text{Cost}\) then break
    end if
    if for all \((l', Z')\) in Passed: \(Z' \not\subseteq Z\) then
        add \((l, Z)\) to Passed
    end if
    add all \((l', Z')\) with \((l, Z) \rightarrow (l', Z')\) to Waiting

return Cost
Experimental Results
## Experiments

### MC Order

<table>
<thead>
<tr>
<th>COST-rates</th>
<th>SCHEDULE</th>
<th>COST</th>
<th>TIME</th>
<th>#Expl</th>
<th>#Pop’d</th>
</tr>
</thead>
<tbody>
<tr>
<td>G&lt; C&lt; BD&gt; C&lt;</td>
<td>60</td>
<td>1762</td>
<td>2638</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G&lt;</td>
<td>55</td>
<td>65</td>
<td>252</td>
<td>378</td>
<td></td>
</tr>
<tr>
<td>GD&gt;</td>
<td>195</td>
<td>65</td>
<td>149</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>CG&gt;</td>
<td>140</td>
<td>60</td>
<td>232</td>
<td>350</td>
<td></td>
</tr>
<tr>
<td>CD&gt;</td>
<td>170</td>
<td>65</td>
<td>263</td>
<td>408</td>
<td></td>
</tr>
<tr>
<td>BD&gt;</td>
<td>975</td>
<td>85</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>-</td>
<td>406</td>
<td>447</td>
<td></td>
</tr>
</tbody>
</table>
Example: Aircraft Landing

- $E$: earliest landing time
- $T$: target time
- $L$: latest time
- $e$: cost rate for being early
- $l$: cost rate for being late
- $d$: fixed cost for being late

Planes have to keep separation distance to avoid turbulences caused by preceding planes.
Example: Aircraft Landing

Planes have to keep separation distance to avoid turbulences caused by preceding planes.
# Aircraft Landing

Source of examples:

Baesley et al’2000

<table>
<thead>
<tr>
<th>problem instance</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of planes</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>30</td>
<td>44</td>
</tr>
<tr>
<td>number of types</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>optimal value</td>
<td>700</td>
<td>1480</td>
<td>820</td>
<td>2520</td>
<td>3100</td>
<td>24442</td>
<td>1550</td>
</tr>
<tr>
<td>explored states</td>
<td>481</td>
<td>2149</td>
<td>920</td>
<td>5693</td>
<td>15069</td>
<td>122</td>
<td>662</td>
</tr>
<tr>
<td>cpu time (secs)</td>
<td>4.19</td>
<td>25.30</td>
<td>11.05</td>
<td>87.67</td>
<td>220.22</td>
<td>0.60</td>
<td>4.27</td>
</tr>
<tr>
<td>optimal value</td>
<td>90</td>
<td>210</td>
<td>60</td>
<td>640</td>
<td>650</td>
<td>554</td>
<td>0</td>
</tr>
<tr>
<td>explored states</td>
<td>1218</td>
<td>1797</td>
<td>669</td>
<td>28821</td>
<td>47993</td>
<td>9035</td>
<td>92</td>
</tr>
<tr>
<td>cpu time (secs)</td>
<td>17.87</td>
<td>39.92</td>
<td>11.02</td>
<td>755.84</td>
<td>1085.08</td>
<td>123.72</td>
<td>1.06</td>
</tr>
<tr>
<td>optimal value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>130</td>
<td>170</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>explored states</td>
<td>24</td>
<td>46</td>
<td>84</td>
<td>207715</td>
<td>189602</td>
<td>62</td>
<td>N/A</td>
</tr>
<tr>
<td>cpu time (secs)</td>
<td>0.36</td>
<td>0.70</td>
<td>1.71</td>
<td>14786.19</td>
<td>12461.47</td>
<td>0.68</td>
<td>N/A</td>
</tr>
<tr>
<td>optimal value</td>
<td>0</td>
<td>0</td>
<td>N/A</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>explored states</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>65</td>
<td>64</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>cpu time (secs)</td>
<td>1.97</td>
<td>1.53</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Branch & Bound Algorithm

Cost := ∞
Passed := ∅
Waiting := {(l₀, Z₀)}

while Waiting ≠ ∅ do
    select (l, Z) from Waiting
    if l = l₀ and minCost(Z) < Cost then
        Cost := minCost(Z)
    if minCost(Z) + Rem(l, Z) ≥ Cost then break
    if for all (l', Z') in Passed, Z' ≠ Z then
        add (l, Z) to Passed
        add all (l', Z') with (l, Z) → (l', Z') to Waiting

return Cost
Zone LP $\rightarrow$ Min Cost Flow

Exploiting duality

\[
\begin{align*}
\text{minimize } & \quad 3x_1 - 2x_2 + 7 \\
\text{when } & \quad x_1 - x_2 \leq 1 \\
& \quad 1 \leq x_2 \leq 3 \\
& \quad x_2 \geq 1
\end{align*}
\]

\[
\begin{align*}
\text{minimize } & \quad 3y_{2,0} - y_{0,2} + y_{1,2} - y_{0,1} \\
\text{when } & \quad y_{2,0} - y_{0,1} - y_{0,2} = 1 \\
& \quad y_{0,2} + y_{1,2} = 2 \\
& \quad y_{0,1} - y_{1,2} = -3
\end{align*}
\]
Zone LP → Min Cost Flow

Exploiting duality

minimize \(3x_1 - 2x_2 + 7\)

when
\[
\begin{align*}
\begin{cases}
x_1 - x_2 & \leq 1 \\
1 & \leq x_2 \leq 3 \\
x_2 & \geq 1
\end{cases}
\end{align*}
\]

minimize \(3y_{2,0} - y_{0,2} + y_{1,2} - y_{0,1}\)

when
\[
\begin{align*}
\begin{cases}
y_{2,0} - y_{0,1} - y_{0,2} & = 1 \\
y_{0,2} + y_{1,2} & = 2 \\
y_{0,1} - y_{1,2} & = -3
\end{cases}
\end{align*}
\]
# Aircraft Landing

*Using MCF/Netsimplex*

<table>
<thead>
<tr>
<th>RW</th>
<th>Planes</th>
<th>Types</th>
<th>10s</th>
<th>15s</th>
<th>20s</th>
<th>20s</th>
<th>20s</th>
<th>30s</th>
<th>44s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>simplex</td>
<td></td>
<td>0.844s</td>
<td>5.210s</td>
<td>2.135s</td>
<td>17.888s</td>
<td>44.878s</td>
<td>0.451s</td>
<td>0.670s</td>
</tr>
<tr>
<td></td>
<td>netsimplex</td>
<td></td>
<td>0.156s</td>
<td>0.657s</td>
<td>0.369s</td>
<td>2.363s</td>
<td>5.503s</td>
<td>0.127s</td>
<td>0.322s</td>
</tr>
<tr>
<td>factor</td>
<td></td>
<td></td>
<td>5.41</td>
<td>7.93</td>
<td>5.79</td>
<td>7.57</td>
<td>8.16</td>
<td>3.55</td>
<td>2.08</td>
</tr>
<tr>
<td></td>
<td>simplex</td>
<td></td>
<td>2.577s</td>
<td>7.436s</td>
<td>2.175s</td>
<td>94.357s</td>
<td>120.004s</td>
<td>2.322s</td>
<td>0.264s</td>
</tr>
<tr>
<td></td>
<td>netsimplex</td>
<td></td>
<td>0.332s</td>
<td>1.036s</td>
<td>0.436s</td>
<td>13.376s</td>
<td>18.033s</td>
<td>0.600s</td>
<td>0.179s</td>
</tr>
<tr>
<td>factor</td>
<td></td>
<td></td>
<td>8.00</td>
<td>7.18</td>
<td>4.99</td>
<td>7.054</td>
<td>6.65</td>
<td>3.87</td>
<td>1.474</td>
</tr>
<tr>
<td></td>
<td>simplex</td>
<td></td>
<td>0.120s</td>
<td>0.181s</td>
<td>0.357s</td>
<td>740.100s</td>
<td>516.678s</td>
<td>0.166s</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>netsimplex</td>
<td></td>
<td>0.064s</td>
<td>0.104s</td>
<td>0.129s</td>
<td>170.176s</td>
<td>124.805s</td>
<td>0.079s</td>
<td>N/A</td>
</tr>
<tr>
<td>factor</td>
<td></td>
<td></td>
<td>1.87</td>
<td>1.74</td>
<td>2.77</td>
<td>4.34</td>
<td>4.14</td>
<td>2.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>simplex</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>1.603s</td>
<td>0.318s</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>netsimplex</td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0.378s</td>
<td>0.093s</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>factor</td>
<td></td>
<td></td>
<td>4.24</td>
<td>3.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Optimal Infinite Scheduling

with Ed Brinksma
Patricia Bouyer
Arne Skou
EXAMPLE: Optimal WORK plan for cars with different subscription rates for city driving!

**Golf**

- Maximal 100 min. at each location

**Citroen**

**BMW**

**Datsun**
Workplan I

\[
\text{Value of workplan:} \quad \frac{4 \times 300}{90} = 13.33
\]
Workplan II

Value of workplan: \[ \frac{560}{100} = 5.6 \]
Infinite Scheduling

\[ E[] \ (Km.x \leq 100 \text{ and } Jacob.x \leq 90 \text{ and } Gerd.x \leq 100) \]
Infinite Optimal Scheduling

\[ E[ \text{ Kim} \cdot x \leq 100 \text{ and Jacob} \cdot x \leq 90 \text{ and Gerd} \cdot x \leq 100 ] \]

+ most minimal limit of cost/time
Cost Optimal Scheduling = Optimal Infinite Path

Value of path $\sigma$: $\text{val}(\sigma) = \lim_{n \to \infty} \frac{c_n}{t_n}$

Optimal Schedule $\sigma^*$: $\text{val}(\sigma^*) = \inf_{\sigma} \text{val}(\sigma)$
Cost Optimal Scheduling = Optimal Infinite Path

\[ \sigma: \quad \text{val}(\sigma) = \lim_{n \to \infty} \frac{c_n}{t_n} \]

Optimal Schedule \( \sigma^* \):

\[ \text{val}(\sigma^*) = \inf_{\sigma} \text{val}(\sigma) \]

\( \neg (\text{Car0.Err or Car1.Err or \ldots}) \)

\( \tau \)

THEOREM: \( \sigma^* \) is computable

Bouyer, Brinksma, Larsen HSCC’04
Application

Dynamic Voltage Scaling
Dynamic Scheduling

E<> ( Kim.Done and Jacob.Done and Gerd.Done )

Cost-Optimal Reachability Strategies for Priced Timed Game Automata
[Alur et al, ICALP’04]
[Bouyer, Cassez. Fleury, Larsen, FSTTCS’04]

Time-Optimal Reachability Strategies for Timed Games
[Cassez, David, Fleury, Larsen, Lime, CONCUR’05]

Uncontrollable timing uncertainty