On the optimal reachability problem in weighted timed games

Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France

Based on former works with Thomas Brihaye, Kim G. Larsen, Nicolas Markey, etc…
And on recent work with Samy Jaziri and Nicolas Markey
Outline

1 Introduction

2 Overview of “old” results
   - Weighted timed automata
   - Timed games
   - Weighted timed games

3 Some recent developments
   - Undecidability of the value problem
   - Approximation of the optimal cost
   - Back to the undecidability

4 Conclusion
Time-dependent systems

- We are interested in timed systems
Time-dependent systems

- We are interested in timed systems
Time-dependent systems

- We are interested in timed systems

- ... and in their analysis and control
An example: The task graph scheduling problem

Compute \( D \times (C \times (A+B)) + (A+B) + (C \times D) \) using two processors:

\[ P_1 \text{ (fast)}: \]

\[
\begin{array}{|c|c|}
\hline
\text{time} & \text{energy} \\
\hline
+ & 2 \text{ picoseconds} \\
\times & 3 \text{ picoseconds} \\
\hline
\end{array}
\]

\[ P_2 \text{ (slow)}: \]

\[
\begin{array}{|c|c|}
\hline
\text{time} & \text{energy} \\
\hline
+ & 5 \text{ picoseconds} \\
\times & 7 \text{ picoseconds} \\
\hline
\end{array}
\]

An example: The task graph scheduling problem

Compute \( D \times (C \times (A+B)) + (A+B) + (C \times D) \) using two processors:

\[ P_1 \text{ (fast):} \]

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th></th>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+ 2 ps</td>
<td></td>
<td>idle</td>
</tr>
<tr>
<td></td>
<td>\times</td>
<td>3 ps</td>
<td>in use</td>
</tr>
</tbody>
</table>

\[ P_2 \text{ (slow):} \]

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th></th>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+ 5 ps</td>
<td></td>
<td>idle</td>
</tr>
<tr>
<td></td>
<td>\times</td>
<td>7 ps</td>
<td>in use</td>
</tr>
</tbody>
</table>

\[ P_1 \quad P_2 \]

[Sch1] \[ T_2 \quad T_3 \quad T_5 \quad T_6 \]

\[ T_1 \quad T_4 \]

\[ P_2 \quad P_1 \]

\[ T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5 \quad T_6 \]

An example: The task graph scheduling problem

Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

$P_1$ (fast):

- Time:
  - $+$: 2 picoseconds
  - $\times$: 3 picoseconds

- Energy:
  - Idle: 10 Watt
  - In use: 90 Watts

$P_2$ (slow):

- Time:
  - $+$: 5 picoseconds
  - $\times$: 7 picoseconds

- Energy:
  - Idle: 20 Watts
  - In use: 30 Watts

An example: The task graph scheduling problem

Compute $D \times (C \times (A+B)) + (A+B) + (C \times D)$ using two processors:

$P_1$ (fast):

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>2 picoseconds</td>
</tr>
<tr>
<td>$\times$</td>
<td>3 picoseconds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Energy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>10 Watt</td>
</tr>
<tr>
<td>in use</td>
<td>90 Watts</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>5 picoseconds</td>
</tr>
<tr>
<td>$\times$</td>
<td>7 picoseconds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Energy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>20 Watts</td>
</tr>
<tr>
<td>in use</td>
<td>30 Watts</td>
</tr>
</tbody>
</table>

$\begin{align*}
D \times (C \times (A+B)) + (A+B) + (C \times D) + T_1 \times T_2 \times T_3 + T_4 \times T_5 + T_6
\end{align*}$

The model of timed automata
The model of timed automata

- **Safe**
  - $x = 0$
  - $y = 0$

- **Alarm**
  - $15 \leq x \leq 16$
  - $2 \leq y \land x \leq 56$
  - $y = 0$

- **Repairing**
  - $2 \leq y \leq 25$
  - $x = 0$
  - $y = 0$

- **Fail-safe**
  - $22 \leq y \leq 25$

### Transition Table

<table>
<thead>
<tr>
<th>State</th>
<th>Next State</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe</td>
<td>Safe</td>
<td>$x = 23$</td>
</tr>
<tr>
<td></td>
<td>Problem</td>
<td>$x = 0$</td>
</tr>
<tr>
<td>Alarm</td>
<td>Alarm</td>
<td>$y = 15.6$</td>
</tr>
<tr>
<td></td>
<td>Delayed</td>
<td>$y = 0$</td>
</tr>
<tr>
<td></td>
<td>Fail-safe</td>
<td>$x = 22$</td>
</tr>
<tr>
<td></td>
<td>Done</td>
<td>$x = 40$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Initial Value</th>
<th>Transition</th>
<th>New Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>0</td>
<td>$x = 23$</td>
<td>23</td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>$y = 23$</td>
<td>23</td>
</tr>
<tr>
<td>$x$</td>
<td>0</td>
<td>$x = 15.6$</td>
<td>15.6</td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>$y = 38.6$</td>
<td>38.6</td>
</tr>
<tr>
<td>$x$</td>
<td>0</td>
<td>$x = 15.6$</td>
<td>15.6</td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>$y = 0$</td>
<td>0</td>
</tr>
<tr>
<td>$x$</td>
<td>0</td>
<td>$x = 22$</td>
<td>22</td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>$y = 0$</td>
<td>0</td>
</tr>
<tr>
<td>$x$</td>
<td>0</td>
<td>$x = 40$</td>
<td>40</td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>$y = 22.1$</td>
<td>22.1</td>
</tr>
<tr>
<td>$x$</td>
<td>0</td>
<td>$x = 40$</td>
<td>40</td>
</tr>
</tbody>
</table>
Modelling the task graph scheduling problem
Modelling the task graph scheduling problem

**Processors**

\[ P_1: \]
\[ + \]
\[ (x \leq 2) \]
\[ x:=0 \]
\[ \text{add}_1 \]
\[ \text{done}_1 \]
\[ \times \]

\[ P_2: \]
\[ + \]
\[ (y \leq 5) \]
\[ x:=0 \]
\[ \text{add}_2 \]
\[ \text{done}_2 \]
\[ \times \]
Modelling the task graph scheduling problem

- **Processors**
  - $P_1$: $x_1 = 2$ \( \text{done}_1 \text{add}_1 \) \( x_1 = 3 \) \( \text{done}_1 \text{mult}_1 \) \( x_1 \leq 2 \) \( x_1 \leq 3 \)
  - $P_2$: $y_2 = 5$ \( \text{done}_2 \text{add}_2 \) \( y_2 = 7 \) \( \text{done}_2 \text{mult}_2 \) \( y_2 \leq 5 \) \( y_2 \leq 7 \)

- **Tasks**
  - $T_4$: $t_4 \leftarrow t_1 \land t_2$ $t_4 = 1$
  - $T_5$: $t_5 \leftarrow t_3$ $t_5 = 1$

A schedule is a path in the product automaton
Analyzing timed automata

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Efficient symbolic technics based on zones, implemented in tools Skip regions.
Analyzing timed automata

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Efficient symbolic technics based on zones, implemented in tools...
Analyzing timed automata

Theorem [AD94] Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction

Efficient symbolic technics based on zones, implemented in tools.
Analyzing timed automata

Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction

Efficient symbolic technics based on zones, implemented in tools.
Analyzing timed automata

Theorem [AD94] Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction

Efficient symbolic techniques based on zones, implemented in tools.
Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.
Technical tool: region abstraction
Efficient symbolic technics based on zones, implemented in tools...
Analyzing timed automata

Theorem [AD94] Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Analyzing timed automata

Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.
Analyzing timed automata

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Efficient symbolic techniques based on zones, implemented in tools.
Analyzing timed automata

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Efficient symbolic technics based on zones, implemented in tools...
Analyzing timed automata

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Efficient symbolic technics based on zones, implemented in tools Skip regions.
Analyzing timed automata

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction
Efficient symbolic technics based on zones, implemented in tools Skip regions
Analyzing timed automata

Theorem [AD94] Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

Technical tool: region abstraction Efficient symbolic technics based on zones, implemented in tools Skip regions
Analyzing timed automata

Theorem [AD94]

Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction
Analyzing timed automata

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction
- Efficient symbolic technics based on zones, implemented in tools
Analyzing timed automata

Theorem [AD94]
Reachability in timed automata is decidable (as well as many other important properties). It is PSPACE-complete.

- Technical tool: region abstraction
- Efficient symbolic technics based on zones, implemented in tools
Technical tool: Region abstraction

\[
\begin{align*}
\text{clock } y & \sim c & \text{with } c \in \{0, 1, 2\} \\
\text{clock } x & \sim c & \text{with } c \in \{0, 1, 2\}
\end{align*}
\]

- The path \(x = 1, y = 1\) can be fired from;
- cannot be fired from.

This is a finite time-abstract bisimulation!
Technical tool: Region abstraction

only constraints: $x \sim c$ with $c \in \{0, 1, 2\}$
$y \sim c$ with $c \in \{0, 1, 2\}$

“compatibility” between regions and constraints
Technical tool: Region abstraction

The path \( x=1 \) - can be fired from
- cannot be fired from

"compatibility" between regions and constraints
"compatibility" between regions and time elapsing
Technical tool: Region abstraction

\[\begin{array}{c}
\text{clock } y \\
\hline
2 \\
1 \\
0 \\
\hline
0 & 1 & 2 \\
\end{array}\]

\[\begin{array}{c}
\text{clock } x \\
\hline
0 & 1 & 2 \\
\end{array}\]

\(\sim\) This is a finite time-abstract bisimulation!
Technical tool: Region abstraction – An example [AD94]
Technical tool: Region abstraction – An example [AD94]
Technical tool: Region abstraction – An example [AD94]
Outline

1 Introduction

2 Overview of “old” results
   • Weighted timed automata
   • Timed games
   • Weighted timed games

3 Some recent developments
   • Undecidability of the value problem
   • Approximation of the optimal cost
   • Back to the undecidability

4 Conclusion
Outline

1 Introduction

2 Overview of “old” results
   - Weighted timed automata
   - Timed games
   - Weighted timed games

3 Some recent developments
   - Undecidability of the value problem
   - Approximation of the optimal cost
   - Back to the undecidability

4 Conclusion
Modelling resources in timed systems

- System *resources* might be relevant and even crucial information
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...

- price to pay,
- bandwidth,
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...

⇒ timed automata are not powerful enough!
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...

  \[\sim\] timed automata are not powerful enough!

- A possible solution: use hybrid automata
  - a discrete control (the mode of the system)
  - continuous evolution of the variables within a mode
Modelling resources in timed systems

- System **resources** might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...

  \[ T \leq 19 \]

  \[ T \geq 21 \]

  \[ T \leq 22 \]

  \[ T \geq 18 \]

→ timed automata are not powerful enough!

- A possible solution: use **hybrid automata**

The thermostat example
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...

→ timed automata are not powerful enough!

- A possible solution: use hybrid automata

The thermostat example

\[ \begin{align*}
\text{Off} & \quad \dot{T} = -0.5T \\
\quad (T \geq 18) & \\
\text{On} & \quad \dot{T} = 2.25 - 0.5T \\
\quad (T \leq 22) & \\
\end{align*} \]
Ok...
Ok...

Easy...
Ok...

Easy...
Ok...

Easy...

Easy...
Ok... but?

Easy... constraint

Easy... constraint
Ok... but?

Easy...

Hard!

constraint

Easy...

constraint
Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...
  \[\rightarrow\] timed automata are not powerful enough!

- A possible solution: use hybrid automata

**Theorem** [HKPV95]

The reachability problem is **undecidable** in hybrid automata. Even for the simplest, the so-called stopwatch automata (clocks can be stopped).

Modelling resources in timed systems

- System resources might be relevant and even crucial information
  - energy consumption,
  - memory usage,
  - ...  
  \[\Rightarrow\] timed automata are not powerful enough!

- A possible solution: use hybrid automata

**Theorem [HKPV95]**

The reachability problem is **undecidable** in hybrid automata. Even for the simplest, the so-called stopwatch automata (clocks can be stopped).

- An alternative: **weighted/priced timed automata** [ALP01,BFH+01]  
  \[\Rightarrow\] hybrid variables do not constrain the system  
  hybrid variables are **observer** variables

Modelling the task graph scheduling problem

- **Processors**
  - $P_1$: $x=2 \Rightarrow \text{add}_1 \Rightarrow \text{done}_1 \Rightarrow \text{idle} \Rightarrow x=0 \Rightarrow x=3 \Rightarrow \text{mult}_1 \Rightarrow \times$
  - $P_2$: $y=5 \Rightarrow \text{add}_2 \Rightarrow \text{done}_2 \Rightarrow \text{idle} \Rightarrow x=0 \Rightarrow y=7 \Rightarrow \times$

- **Tasks**
  - $T_4$: $t_1 \land t_2 \Rightarrow \text{add}_i \Rightarrow t_4:=1 \Rightarrow \text{done}_i$
  - $T_5$: $t_3 \Rightarrow \text{add}_i \Rightarrow t_5:=1 \Rightarrow \text{done}_i$
Modelling the task graph scheduling problem

- **Processors**
  - $P_1$: 
    - $(x \leq 2)$
    - $x := 0$
    - $x := 0$
    - $x := 0$
  - $P_2$: 
    - $(y \leq 5)$
    - $x := 0$
    - $x := 0$
    - $x := 0$

- **Tasks**

- **Modelling energy**
  - $P_1$: 
    - $(x \leq 2)$
    - $x := 0$
    - $x := 0$
  - $P_2$: 
    - $(y \leq 5)$
    - $x := 0$
    - $x := 0$

A good schedule is a path in the product automaton with a low cost.
Weighted/priced timed automata [ALP01,BFH+01]

\[
\begin{align*}
\ell_0 & \xrightarrow{x \leq 2, c, y := 0} \ell_1 \\
\ell_1 & \xrightarrow{u} \ell_2 \quad \ell_2 \\
\ell_2 & \xrightarrow{x = 2, c} +1 \\
\ell_2 & \xrightarrow{u} \ell_3 \quad \ell_3 \\
\ell_3 & \xrightarrow{x = 2, c} +1
\end{align*}
\]

Weighted/priced timed automata \cite{ALP01,BFH+01}

\[ x \leq 2, c, y := 0 \]

\[
\begin{align*}
\ell_0 & \xrightarrow{1.3} \ell_0 \\
\ell_0 & \xrightarrow{c} \ell_1 \\
\ell_1 & \xrightarrow{u} \ell_3 \\
\ell_3 & \xrightarrow{0.7} \ell_3 \\
\ell_3 & \xrightarrow{c} \smiley
\end{align*}
\]

\[
\begin{array}{c|c|c|c|c|c}
\text{cost} & \ell_0 & \ell_0 & \ell_1 & \ell_3 & \ell_3 \\
\hline
x & 0 & 1.3 & 1.3 & 1.3 & 2 \\
y & 0 & 1.3 & 0 & 0 & 0.7 \\
\end{array}
\]

\cite{ALP01} Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC’01).

\cite{BFH+01} Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (HSCC’01).
**Weighted/priced timed automata [ALP01,BFH+01]**

![Diagram](image)

\[
\begin{align*}
\ell_0 & \xrightarrow{1.3} \ell_0 & \ell_0 & \xrightarrow{c} \ell_1 & \ell_1 & \xrightarrow{u} \ell_3 & \ell_3 & \xrightarrow{0.7} \ell_3 & \ell_3 & \xrightarrow{c} \text{smiley face}
\end{align*}
\]

\[
\begin{align*}
x & = 0 & 1.3 & 1.3 & u & 1.3 & 1.3 & 2 & c & 0.7
\end{align*}
\]

\[
cost: 6.5 + 0 + 0 + 0.7 = 14.2
\]

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata (*HSCC'01*).

[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (*HSCC'01*).
Weighted/priced timed automata [ALP01,BFH+01]

\[
\begin{array}{cccccc}
\ell_0 & \overset{1.3}{\rightarrow} & \ell_0 & \overset{c}{\rightarrow} & \ell_1 & \overset{u}{\rightarrow} & \ell_3 & \overset{0.7}{\rightarrow} & \ell_3 & \overset{c}{\rightarrow} & \text{smile} \\
x & 0 & 1.3 & 1.3 & 1.3 & 1.3 & 2 & 0.7 & 0.7 & 0.7 & \text{cost : } 6.5 \\
y & 0 & 1.3 & 0 & 0 & 0 & 0.7 & & & & \\
\end{array}
\]

Weighted/priced timed automata [ALP01,BFH+01]

\[
\begin{align*}
\ell_0 & \xrightarrow{1.3} \ell_0 \\
x & = 0 \\
y & = 1.3 \\
c & \rightarrow \\
x & \leq 2, c, y := 0 \\
\ell_1 & \rightarrow \\
\ell_2 & \rightarrow \\
x & = 2, c \\
\ell_3 & \rightarrow \\
P & = 10 \\
x & = 2, c \\
\ell_4 & \rightarrow \\
\ell_5 & \rightarrow \\
x & = 2, c \\
+1 & \\
\end{align*}
\]

- Cost: \(6.5 + 0\)

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).

Weighted/priced timed automata [ALP01,BFH+01]

\[
\begin{array}{cccccc}
\ell_0 & \xrightarrow{1.3} & \ell_0 & \xrightarrow{c} & \ell_1 & \xrightarrow{u} & \ell_3 & \xrightarrow{0.7} & \ell_3 & \xrightarrow{c} & \text{宯} \\
0 & 1.3 & 1.3 & 1.3 & 1.3 & 2 & 0.7 & & & & \\
\end{array}
\]

\[
\begin{array}{c}
x \leq 2, \ c, \ y := 0 \\
x \leq 2, \ c \\
x = 2, \ c \\
x = 2, \ c \\
\end{array}
\]

\[
\begin{align*}
\ell_0 & \rightarrow \ell_1 & \ell_1 & \rightarrow \ell_2 & \ell_2 & \rightarrow \ell_3 & \ell_3 & \rightarrow \text{宯} \\
5 & \rightarrow \ell_1 & \ell_1 & \rightarrow \ell_2 & \ell_2 & \rightarrow \ell_3 & \ell_3 & \rightarrow \text{宯} \\
\end{align*}
\]

\[
\begin{align*}
\ell_0 & \rightarrow \ell_1 & \ell_1 & \rightarrow \ell_2 & \ell_2 & \rightarrow \ell_3 & \ell_3 & \rightarrow \text{宯} \\
5 & \rightarrow \ell_1 & \ell_1 & \rightarrow \ell_2 & \ell_2 & \rightarrow \ell_3 & \ell_3 & \rightarrow \text{宯} \\
\end{align*}
\]

\[
\begin{align*}
\text{cost} & : 6.5 + 0 + 0 \\
\end{align*}
\]

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).

Weighted/priced timed automata \cite{ALP01,BFH+01}

\[
\begin{align*}
\ell_0 & \xrightarrow{1.3} \ell_0 & \xrightarrow{c} \ell_1 & \xrightarrow{u} \ell_3 & \xrightarrow{0.7} \ell_3 & \xrightarrow{c} & \smiley
\end{align*}
\]

\[
\begin{array}{cccccc}
\ell_0 & \ell_0 & \ell_1 & \ell_3 & \ell_3 & \smiley \\
\ell_0 & 1.3 & 1.3 & 1.3 & 2 & \\
\ell_1 & 1.3 & 0 & 0 & 0.7 & \\
x & 0 & 1.3 & 1.3 & 1.3 & 2 \\
y & 0 & 1.3 & 0 & 0 & 0.7 \\
\text{cost :} & 6.5 & + & 0 & + & 0 & + & 0.7
\end{array}
\]

\cite{ALP01} Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC’01).
\cite{BFH+01} Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (HSCC’01).
Weighted/priced timed automata \[\text{[ALP01,BFH+01]}\]

\[\begin{array}{cccccccc}
\ell_0 & \xrightarrow{1.3} & \ell_0 & \xrightarrow{c} & \ell_1 & \xrightarrow{u} & \ell_3 & \xrightarrow{0.7} & \ell_3 & \xrightarrow{c} & \smiley \\
0 & 1.3 & 1.3 & 1.3 & 2 & 0.7 & \end{array}\]

\[
\begin{align*}
\ell_0 & \rightarrow x \leq 2, c, y := 0 \\
\ell_1 & \rightarrow (y=0) \\
\ell_2 & \rightarrow x=2, c \rightarrow +10 \\
\ell_3 & \rightarrow +1 \\
\ell_4 & \rightarrow +7
\end{align*}
\]

\[
\text{cost: } 6.5 + 0 + 0 + 0.7 + 7
\]


Weighted/priced timed automata \([\text{ALP01,BFH+01]}\)

\[
\begin{align*}
\ell_0 & \overset{\ell_0}{\overset{1.3}{\rightarrow}} \ell_0 \overset{c}{\rightarrow} \ell_1 \overset{u}{\rightarrow} \ell_3 \overset{0.7}{\rightarrow} \ell_3 \overset{c}{\rightarrow} \text{\smile} \\
x & 0 \quad 1.3 \quad 1.3 \quad 1.3 \quad 2 \\
y & 0 \quad 1.3 \quad 0 \quad 0 \quad 0.7 \\
\text{cost} & : \quad 6.5 \quad + \quad 0 \quad + \quad 0 \quad + \quad 0.7 \quad + \quad 7 \quad = \quad 14.2
\end{align*}
\]

[\text{ALP01}] \ Alur, La Torre, Pappas. Optimal paths in weighted timed automata (HSCC'01).

[\text{BFH+01}] \ Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced timed automata (HSCC'01).
Weighted/priced timed automata [ALP01, BFH+01]

**Question:** what is the optimal cost for reaching 🎉?


Weighted/priced timed automata [ALP01,BFH+01]

**Question:** what is the optimal cost for reaching \( \smiley \)?

\[
5t + 10(2 - t) + 1
\]


Weighted/priced timed automata [ALP01,BFH+01]

Question: what is the optimal cost for reaching ☺?

\[ 5t + 10(2 - t) + 1, \quad 5t + (2 - t) + 7 \]


Weighted/priced timed automata [ALP01,BFH+01]

Question: what is the optimal cost for reaching 😊?

\[
\min \left( 5t + 10(2 - t) + 1, \ 5t + (2 - t) + 7 \right)
\]

Weighted/priced timed automata \[\text{[ALP01,BFH+01]}\]

**Question:** what is the optimal cost for reaching \(\smiley\) ?

\[
\inf_{0 \leq t \leq 2} \min \left( 5t + 10(2 - t) + 1 , \ 5t + (2 - t) + 7 \right) = 9
\]
Weighted/priced timed automata [ALP01,BFH+01]

**Question:** what is the optimal cost for reaching 😊?

\[
\inf_{0 \leq t \leq 2} \min (5t + 10(2 - t) + 1, 5t + (2 - t) + 7) = 9
\]

\[\sim strategy: \text{ leave immediately } \ell_0, \text{ go to } \ell_3, \text{ and wait there } 2 \text{ t.u.}\]

Optimal-cost reachability

**Theorem [ALP01,BFH+01,BBBR07]**

In weighted timed automata, the optimal cost is an integer and can be computed in PSPACE.

- Technical tool: a refinement of the regions, the corner-point abstraction

[ALP01] Alur, La Torre, Pappas. Optimal paths in weighted timed automata (*HSCC’01*).
From timed to discrete behaviours

Optimal reachability as a linear programming problem
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[
\begin{align*}
  t_1 & \quad t_2 & \quad t_3 & \quad t_4 & \quad t_5 & \quad \ldots \\
\end{align*}
\]
From timed to discrete behaviours

**Optimal reachability as a linear programming problem**

\[ t_1 + t_2 \leq 2 \]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[
\begin{align*}
\text{t}_1 & \xrightarrow{y:=0} \text{t}_2 \xrightarrow{x\leq 2} \text{t}_3 \xrightarrow{y\geq 5} \text{t}_4 & \xrightarrow{t_5} \cdots \\
& \phantom{\text{t}_1} & \phantom{\text{t}_2} & \phantom{\text{t}_3} & \phantom{\text{t}_4} \text{t}_1 + \text{t}_2 \leq 2 \\
& & & \phantom{\text{t}_4} \phantom{\text{t}_5} \text{t}_2 + \text{t}_3 + \text{t}_4 \geq 5
\end{align*}
\]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[ \begin{align*}
T_1 & \quad T_2 & \quad T_3 & \quad T_4 & \quad T_5 \\
\quad t_1 & \quad t_2 & \quad t_3 & \quad t_4 & \quad t_5 & \quad \ldots
\end{align*} \]

\[ \begin{align*}
y := 0 & \quad x \leq 2 & \quad y \geq 5 & \quad T_2 \leq 2 \\
t_1 + t_2 & \leq 2 & \quad t_2 + t_3 + t_4 & \geq 5 & \quad T_4 - T_1 & \geq 5
\end{align*} \]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[
\begin{align*}
T_1 & \quad T_2 & \quad T_3 & \quad T_4 & \quad T_5 \\
\circ & \quad \xrightarrow{t_1} \quad \circ & \quad \xrightarrow{t_2} \quad \circ & \quad \xrightarrow{t_3} \quad \circ & \quad \xrightarrow{t_4} \quad \circ & \quad \xrightarrow{t_5} \quad \circ & \quad \cdots
\end{align*}
\]

\[
\begin{align*}
y := 0 & \quad x \leq 2 & \quad y \geq 5
\end{align*}
\]

\[
\begin{align*}
t_1 + t_2 & \leq 2 & \quad T_2 \leq 2 \\
t_2 + t_3 + t_4 & \geq 5 & \quad T_4 - T_1 & \geq 5
\end{align*}
\]

Lemma

Let \( Z \) be a bounded zone and \( f \) be a function

\[
f : (T_1, \ldots, T_n) \mapsto \sum_{i=1}^{n} c_i T_i + c
\]

well-defined on \( \overline{Z} \). Then \( \inf_Z f \) is obtained on the border of \( \overline{Z} \) with integer coordinates.
From timed to discrete behaviours

**Optimal reachability as a linear programming problem**

\[
T_1 \quad T_2 \quad T_3 \quad T_4 \quad T_5
\]

\[
\circ \quad t_1 \quad \circ \quad t_2 \quad \circ \quad t_3 \quad \circ \quad t_4 \quad \circ \quad t_5 \quad \circ \quad \ldots
\]

\[
y := 0 \quad x \leq 2 \quad y \geq 5\]

\[
t_1 + t_2 \leq 2 \quad T_2 \leq 2
\]
\[
t_2 + t_3 + t_4 \geq 5 \quad T_4 - T_1 \geq 5
\]

**Lemma**

Let \( Z \) be a bounded zone and \( f \) be a function

\[
f : (T_1, \ldots, T_n) \mapsto \sum_{i=1}^{n} c_i T_i + c
\]

well-defined on \( \overline{Z} \). Then \( \inf_{\overline{Z}} f \) is obtained on the border of \( \overline{Z} \) with integer coordinates.

\( \leadsto \) for every finite path \( \pi \) in \( \mathcal{A} \), there exists a path \( \Pi \) in \( \mathcal{A}_{cp} \) such that

\[
\text{cost}(\Pi) \leq \text{cost}(\pi)
\]

[\( \Pi \) is a “corner-point projection” of \( \pi \)]
Approximation of abstract paths:

For any path $\Pi$ of $A_{cp}$,
From discrete to timed behaviours

**Approximation of abstract paths:**

For any path $\Pi$ of $A_{cp}$, for any $\varepsilon > 0$,
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $\mathcal{A}_{cp}$, for any $\varepsilon > 0$, there exists a path $\pi_\varepsilon$ of $\mathcal{A}$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $A_{cp}$, for any $\varepsilon > 0$, there exists a path $\pi_\varepsilon$ of $A$ s.t.

$$\|\Pi - \pi_\varepsilon\|_{\infty} < \varepsilon$$

For every $\eta > 0$, there exists $\varepsilon > 0$ s.t.

$$\|\Pi - \pi_\varepsilon\|_{\infty} < \varepsilon \Rightarrow |\text{cost}(\Pi) - \text{cost}(\pi_\varepsilon)| < \eta$$
Note on the corner-point abstraction

It is a very interesting abstraction, that can be used in several other contexts:

- for mean-cost optimization [BBL04,BBL08]
- for discounted-cost optimization [FL08]
- for all concavely-priced timed automata [JT08]
- for deciding frequency objectives [BBBS11,Sta12]
- ...
Going further 1: mean-cost optimization

\[
\begin{align*}
\dot{C} &= p \\
\dot{R} &= g \\
\end{align*}
\]

\[
\begin{align*}
(x \leq D) \\
\dot{x} &= 0 \\
\end{align*}
\]

\[
\begin{align*}
x &= D \\
\end{align*}
\]

\[
\begin{align*}
\dot{C} &= p \\
\dot{R} &= g \\
\end{align*}
\]

\[
\begin{align*}
x &= 0 \\
\end{align*}
\]

\[
\begin{align*}
\text{att?} \\
\text{att?} \\
\text{att!} \\
\text{att!} \\
\text{Op} \\
\text{Op} \\
\end{align*}
\]

\[
\begin{align*}
z &\geq S \\
z &= 0 \\
\end{align*}
\]

Going further 1: mean-cost optimization

\[ \dot{C} = p \]
\[ \dot{R} = g \]

\[ (x \leq D) \]
\[ \dot{x} = 0 \]
\[ x = D \]
\[ \text{att?} \]

\[ att?, x := 0 \]

\[ \text{High} \]
\[ \text{Low} \]
\[ \dot{C} = p \]
\[ \dot{R} = g \]
\[ \text{Op} \]

\[ z \geq S \]
\[ z := 0 \]

\[ \text{att!} \]

compute optimal infinite schedules that minimize

\[ \text{mean-cost}(\pi) = \lim_{n \to +\infty} \sup \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)} \]

Going further 1: mean-cost optimization

\[ \dot{C} = p \quad \dot{R} = g \]

\[ \dot{C} = p \quad \dot{R} = g \]

\[ x = 0 \]

\[ \text{att?} \]

\[ \text{att!} \]

\[ z \geq S \]

\[ z := 0 \]

\[ x \leq D \]

\[ \dot{x} = 0 \]

\[ \text{Op} \]

\[ \text{Schedule with ratio } \approx 1.455 \]

\[ \text{Schedule with ratio } \approx 1.478 \]

\[ \text{mean-cost}(\pi) = \lim_{n \to +\infty} \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)} \]

\[ \text{compute optimal infinite schedules that minimize} \]

\[ \text{[BBL08]} \text{ Bouyer, Brinksma, Larsen. Optimal infinite scheduling for multi-priced timed automata (Formal Methods in System Designs).} \]
Going further 1: mean-cost optimization

\[ \dot{C} = p \quad \dot{R} = g \]

\[ (x \leq D) \quad \dot{x} = 0 \]

\[ \text{High} \rightarrow \text{Low} \]

\[ \text{Low} \rightarrow \text{Op} \]

\[ \text{Op} \rightarrow \text{att!} \quad z \geq S \quad z := 0 \]

\[ \text{att?} \quad x := 0 \]

\[ x = D \]

\[ \text{High} \]

\[ \text{Low} \]

\[ \text{Op} \]

\[ \sim \text{compute optimal infinite schedules that minimize} \]

\[ \text{mean-cost}(\pi) = \lim_{n \to +\infty} \sup \frac{\text{cost}(\pi_n)}{\text{reward}(\pi_n)} \]

**Theorem [BBL08]**

In weighted timed automata, the optimal mean-cost can be compute in PSPACE.

\[ \sim \text{the corner-point abstraction can be used} \]

From timed to discrete behaviours

- **Finite behaviours:** based on the following property

**Lemma**

Let $Z$ be a bounded zone and $f$ be a function

$$f : (t_1, \ldots, t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $\inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.
From timed to discrete behaviours

- **Finite behaviours**: based on the following property

**Lemma**

Let $Z$ be a bounded zone and $f$ be a function

$$f : (t_1, \ldots, t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $Z$. Then $\inf_Z f$ is obtained on the border of $Z$ with integer coordinates.

\[
\leadsto \text{for every finite path } \pi \text{ in } A, \text{ there exists a path } \Pi \text{ in } A_{cp} \text{ s.t.} \\
\text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi)
\]
From timed to discrete behaviours

- **Finite behaviours**: based on the following property

**Lemma**

Let $Z$ be a bounded zone and $f$ be a function

$$f : (t_1, \ldots, t_n) \mapsto \sum_{i=1}^{n} c_i t_i + c - \sum_{i=1}^{n} r_i t_i + r$$

well-defined on $\overline{Z}$. Then $\inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

$\leadsto$ for every finite path $\pi$ in $\mathcal{A}$, there exists a path $\Pi$ in $\mathcal{A}_{cp}$ s.t.

$$\text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi)$$

- **Infinite behaviours**: decompose each sufficiently long projection into cycles:

The (acyclic) linear part will be negligible!
From timed to discrete behaviours

- **Finite behaviours:** based on the following property

**Lemma**

Let $Z$ be a bounded zone and $f$ be a function

$$f : (t_1, \ldots, t_n) \mapsto \sum_{i=1}^{n} c_i t_i + c$$

$$\sum_{i=1}^{n} r_i t_i + r$$

well-defined on $\overline{Z}$. Then $\inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

$\leadsto$ for every finite path $\pi$ in $A$, there exists a path $\Pi$ in $A_{cp}$ s.t.

$$\text{mean-cost}(\Pi) \leq \text{mean-cost}(\pi)$$

- **Infinite behaviours:** decompose each sufficiently long projection into cycles:

The (acyclic) linear part will be negligible!

$\leadsto$ the optimal cycle of $A_{cp}$ is better than any infinite path of $A$!
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $A_{cp}$,
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $A_{cp}$, for any $\varepsilon > 0$, 

\[ \| \Pi - \pi_{\varepsilon} \|_{\infty} < \varepsilon \]

For every $\eta > 0$, there exists $\varepsilon > 0$ s.t.

\[ \| \Pi - \pi_{\varepsilon} \|_{\infty} < \varepsilon \Rightarrow |\text{mean-cost}(\Pi) - \text{mean-cost}(\pi_{\varepsilon})| < \eta \]
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $A_{cp}$, for any $\varepsilon > 0$, there exists a path $\pi_\varepsilon$ of $A$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$
From discrete to timed behaviours

Approximation of abstract paths:

For any path $\Pi$ of $A_{cp}$, for any $\varepsilon > 0$, there exists a path $\pi_\varepsilon$ of $A$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon$$

For every $\eta > 0$, there exists $\varepsilon > 0$ s.t.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon \Rightarrow |\text{mean-cost}(\Pi) - \text{mean-cost}(\pi_\varepsilon)| < \eta$$
Going further 2: concavely-priced cost functions

\[ \leadsto \text{A general abstract framework for quantitative timed systems} \]

**Theorem [JT08]**

In concavely-priced timed automata, optimal cost is computable, if we restrict to quasi-concave cost functions. For the following cost functions, the (decision) problem is even PSPACE-complete:

- optimal-time and optimal-cost reachability;
- optimal discrete discounted cost;
- optimal mean-cost.

\[ \leadsto \text{the corner-point abstraction can be used} \]

[JT08] Judziński, Trivedi. Concavely-priced timed automata (*FORMATS'08*).
Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

```plaintext
& (x \leq 3) & (x \leq 3) & x = 3
\hline
High & deg; +2 & \text{att; } -2 & z \geq 2, x, z := 0
\hline
Med & deg; +5 & \text{att; } +1 & z \geq 2, z := 0
\hline
Low & +9
```

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata (INFINITY'08).
Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

\[
\begin{align*}
\text{High} & \quad \quad \text{deg} \quad \quad \text{att} \quad \quad \text{Low} \\
\text{Med} & \quad \quad \text{deg} \quad \quad \text{att}
\end{align*}
\]

\(x = 3, x := 0\)

\(z \geq 2, x, z := 0\)

\(z \geq 2, z := 0\)

\(x = 3\)

\(x = 3\)

\[
\begin{align*}
\text{High} & \quad +2 \\
\text{Med} & \quad +5 \\
\text{Low} & \quad +9
\end{align*}
\]

\(\sim\) compute optimal infinite schedules that minimize discounted cost over time

Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

\[
\begin{align*}
&x = 3, x := 0 \\
&\text{deg} \\
&\text{att} \\
&z \geq 2, x, z := 0
\end{align*}
\]

\[
\begin{align*}
&x = 3 \\
&\text{deg} \\
&\text{att} \\
&z \geq 2, z := 0
\end{align*}
\]

\(\sim\) compute optimal infinite schedules that minimize

\[
\text{discounted-cost}_\lambda(\pi) = \sum_{n \geq 0} \lambda^{T_n} \int_{t=0}^{\tau_{n+1}} \lambda^t \text{cost}(\ell_n) \, dt + \lambda^{T_{n+1}} \text{cost}(\ell_n \xrightarrow{a_{n+1}} \ell_{n+1})
\]

if \(\pi = (\ell_0, \nu_0) \xrightarrow{\tau_1, a_1} (\ell_1, \nu_1) \xrightarrow{\tau_2, a_2} \cdots\) and \(T_n = \sum_{i \leq n} \tau_i\)

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata (INFINITY'08).
Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

\[
\begin{align*}
&\text{High} \\
\xrightarrow{\text{att}} & z \geq 2, x, z := 0 \quad \xleftarrow{\text{deg}}
\end{align*}
\]

\[
\begin{align*}
&\text{Med} \\
\xrightarrow{\text{att}} & z \geq 2, x, z := 0 \quad \xleftarrow{\text{deg}}
\end{align*}
\]

\[
\begin{align*}
&\text{Low} \\
\xrightarrow{\text{att}} & z \geq 2, x, z := 0 \quad \xleftarrow{\text{deg}}
\end{align*}
\]

\[
\begin{align*}
&x = 3, x := 0 \\
\quad +2
\end{align*}
\]

\[
\begin{align*}
&x = 3 \\
\quad +5
\end{align*}
\]

\[
\begin{align*}
&x = 3 \\
\quad +9
\end{align*}
\]

\[\sim \text{ compute optimal infinite schedules that minimize discounted cost over time} \]

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata (INFINITY'08).
Going further 3: discounted-time cost optimization

Globally, \((z\leq 8)\)

\[
\begin{align*}
& (x\leq 3) \\
& \xrightarrow{\text{High}} \\
& \xleftarrow{\text{High}} \\
& \xrightarrow{\text{Med}} \\
& \xleftarrow{\text{Med}} \\
& \xrightarrow{\text{Low}} \\
& \xleftarrow{\text{Low}} \\
& x=3, x:=0 \\
& \text{deg} \\
& \text{att} \\
& z\geq 2, x, z:=0 \\
& +2 \\
& \xrightarrow{\text{High}} \\
& \xleftarrow{\text{High}} \\
& \xrightarrow{\text{Med}} \\
& \xleftarrow{\text{Med}} \\
& \xrightarrow{\text{Low}} \\
& \xleftarrow{\text{Low}} \\
& x=3 \\
& \text{deg} \\
& \text{att} \\
& z\geq 2, z:=0 \\
& +1 \\
& +9 \\
& +5 \\
& +2 \\
& +1
\end{align*}
\]

\[\sim \quad \text{compute optimal infinite schedules that minimize discounted cost over time}\]

if \(\lambda = e^{-1}\), the discounted cost of that infinite schedule is \(\approx 2.16\)

\[\begin{align*}
& \text{if } \lambda = e^{-1}, \text{ the discounted cost of that infinite schedule is } \approx 2.16
\end{align*}\]

Going further 3: discounted-time cost optimization

Globally, \((z \leq 8)\)

\begin{align*}
(x \leq 3) & \quad \text{High} \quad +2 \quad z \geq 2, x, z := 0 \\
& \quad \text{deg} \quad -2 \quad x = 3, x := 0 \\
& \quad \text{att} \quad +2 \quad z \geq 2, x, z := 0
\end{align*}

\begin{align*}
(x \leq 3) & \quad \text{Low} \quad +9 \quad z \geq 2, z := 0 \\
& \quad \text{att} \quad +1 \quad x = 3 \\
& \quad \text{deg} \quad +5 \quad z \geq 2, x, z := 0
\end{align*}

\(\sim\) compute optimal infinite schedules that minimize discounted cost over time

**Theorem** [FL08]

In weighted timed automata, the optimal discounted cost is computable in EXPTIME.

\(\sim\) the corner-point abstraction can be used

[FL08] Fahrenberg, Larsen. Discount-optimal infinite runs in priced timed automata (*INFINITY'08*).
Outline

1 Introduction

2 Overview of “old” results
   - Weighted timed automata
   - Timed games
   - Weighted timed games

3 Some recent developments
   - Undecidability of the value problem
   - Approximation of the optimal cost
   - Back to the undecidability

4 Conclusion
Modelling the task graph scheduling problem

- **Processors**
  - \( P_1: \)
    - \( x=2 \)
    - \( x:=0 \)
    - \( x=3 \)
    - \( x:=0 \)
    - \( + \) \((x \leq 2)\)
    - \( - \) \((x \leq 3)\)
    - \( \text{idle} \)
    - \( \text{done}_1 \)
    - \( \text{add}_1 \)
    - \( \text{mult}_1 \)
  - \( P_2: \)
    - \( x=2 \)
    - \( x:=0 \)
    - \( x=3 \)
    - \( x:=0 \)
    - \( + \) \((y \leq 5)\)
    - \( - \) \((y \leq 7)\)
    - \( \text{idle} \)
    - \( \text{add}_2 \)
    - \( \text{done}_2 \)
    - \( \text{mult}_2 \)

- **Tasks**
  - \( T_4: \)
    - \( t_1 \land t_2 \)
    - \( t_4 := 1 \)
    - \( \text{add}_i \)
    - \( \text{done}_i \)
  - \( T_5: \)
    - \( t_3 \)
    - \( t_5 := 1 \)
    - \( \text{add}_i \)
    - \( \text{done}_i \)

- **Modelling energy**
  - \( P_1: \)
    - \( x=2 \)
    - \( x:=0 \)
    - \( x=3 \)
    - \( x:=0 \)
    - \( +90 \) \((x \leq 2)\)
    - \( +10 \) \((x \leq 3)\)
    - \( \text{idle} \)
    - \( \text{done}_1 \)
    - \( \text{add}_1 \)
    - \( \text{mult}_1 \)
  - \( P_2: \)
    - \( x=2 \)
    - \( x:=0 \)
    - \( x=3 \)
    - \( x:=0 \)
    - \( +30 \) \((y \leq 5)\)
    - \( +20 \) \((y \leq 7)\)
    - \( \text{idle} \)
    - \( \text{add}_2 \)
    - \( \text{done}_2 \)
    - \( \text{mult}_2 \)
Modelling the task graph scheduling problem

- **Processors**
  - \( P_1 \):
    - \( x = 2 \)
    - \( x = 3 \)
    - \( x = 0 \)
  - \( P_2 \):
    - \( y = 5 \)
    - \( y = 7 \)
    - \( y = 0 \)

- **Tasks**
  - \( T_4 \):
    - \( t_1 \land t_2 \)
    - \( t_4 := 1 \)
  - \( T_5 \):
    - \( t_3 \)
    - \( t_5 := 1 \)

- **Modelling energy**
  - \( P_1 \):
    - \( x = 2 \)
    - \( x = 3 \)
    - \( x = 0 \)
  - \( P_2 \):
    - \( y = 5 \)
    - \( y = 7 \)
    - \( y = 0 \)

- **Modelling uncertainty**
  - \( P_1 \):
    - \( x \geq 1 \)
    - \( x \geq 1 \)
    - \( x = 0 \)
  - \( P_2 \):
    - \( y \geq 3 \)
    - \( y \geq 2 \)
    - \( y = 0 \)
Modelling the task graph scheduling problem

- **Processors**
  - $P_1$: $x = 2$ 
  - $P_2$: $y = 5$

- **Tasks**
  - $T_4$: $t_1 \land t_2$
  - $T_5$: $t_3$

- **Modelling energy**
  - $P_1$: $x = 2$ 
  - $P_2$: $y = 5$

- **Modelling uncertainty**
  - $P_1$: $x \geq 1$
  - $P_2$: $y \geq 3$

A (good) schedule is a strategy in the product game (with a low cost)
An example of a timed game

Rule of the game

- **Aim**: avoid 😞 and reach 😊
An example of a timed game

Rule of the game
- **Aim**: avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

\[
\begin{align*}
(x \leq 2) \\
\ell_0 & \quad \xrightarrow{(x \leq 2)} \ell_1 \\
\ell_1 & \quad \xrightarrow{x \geq 1, u_3} \ell_0 \\
\ell_1 & \quad \xrightarrow{x \leq 1, c_1} \ell_2 \\
\ell_2 & \quad \xrightarrow{x < 1, u_2, x := 0} \ell_3 \\
\ell_2 & \quad \xrightarrow{x < 1, u_1} \ell_1 \\
\ell_2 & \quad \xrightarrow{x \leq 1, c_3} \ell_3 \\
\ell_3 & \quad \xrightarrow{x \geq 2, c_4} \ell_1 \\
\end{align*}
\]
An example of a timed game

Rule of the game

- **Aim**: avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

  \[ f : \text{history} \mapsto (\text{delay}, \text{cont. transition}) \]
An example of a timed game

Rule of the game

- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

\[ f : \text{history} \mapsto (\text{delay}, \text{cont. transition}) \]

A (memoryless) winning strategy

- from \((\ell_0, 0)\), play \((0.5, c_1)\)
An example of a timed game

Rule of the game

- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

\[ f : \text{history} \mapsto (\text{delay, cont. transition}) \]

A (memoryless) winning strategy

- from \((\ell_0, 0)\), play \((0.5, c_1)\)
  \[ \leadsto \text{can be preempted by } u_2 \]
An example of a timed game

Rule of the game
- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:
  
  \[ f : \text{history} \mapsto (\text{delay}, \text{cont. transition}) \]

A (memoryless) winning strategy
- from \((\ell_0, 0)\), play \((0.5, c_1)\)
  - can be preempted by \(u_2\)
- from \((\ell_2, \star)\), play \((1 - \star, c_2)\)
An example of a timed game

**Rule of the game**
- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

  \[ f : \text{history} \mapsto (\text{delay, cont. transition}) \]

**A (memoryless) winning strategy**
- from \((\ell_0, 0)\), play \((0.5, c_1)\)
  - can be preempted by \(u_2\)
- from \((\ell_2, \star)\), play \((1 - \star, c_2)\)
- from \((\ell_3, 1)\), play \((0, c_3)\)
An example of a timed game

Rule of the game

- **Aim:** avoid 🙁 and reach 🙂
- **How do we play?** According to a strategy:

  \[ f : \text{history} \mapsto (\text{delay}, \text{cont. transition}) \]

A (memoryless) winning strategy

- from \((\ell_0, 0)\), play \((0.5, c_1)\) 
  ~ can be preempted by \(u_2\)
- from \((\ell_2, *)\), play \((1 - *, c_2)\)
- from \((\ell_3, 1)\), play \((0, c_3)\)
- from \((\ell_1, 1)\), play \((1, c_4)\)

\(\ell_0\) \(x \leq 2\) \(x \geq 1, u_3\) 
\(\ell_0\) 
\(\ell_1\) \(x \leq 1, c_1\) \(x \geq 1, c_4\) 
\(\ell_1\) 
\(\ell_2\) \(x < 1, u_1\) \(x < 1, u_2, x := 0\) 
\(\ell_2\) \(c_2\) 
\(\ell_3\)
An example of a timed game

Rule of the game
- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:
  \[ f : \text{history} \mapsto (\text{delay}, \text{cont. transition}) \]

Problems to be considered
An example of a timed game

Rule of the game
- **Aim**: avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

\[ f : \text{history} \mapsto (\text{delay, cont. transition}) \]

Problems to be considered
- Does there exist a winning strategy?
An example of a timed game

Rule of the game
- **Aim**: avoid 😞 and reach 😊
- **How do we play?** According to a strategy:
  \[ f : \text{history} \mapsto (\text{delay, cont. transition}) \]

Problems to be considered
- Does there exist a winning strategy?
- If yes, compute one (as simple as possible).
Decidability of timed games

**Theorem [AMPS98,HK99]**

Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and “region-based” strategies are sufficient.


[HK99] Henzinger, Kopke. Discrete-time control for rectangular hybrid automata (*Theoretical Computer Science*).
Decidability of timed games

**Theorem** [AMPS98,HK99]

Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and “region-based” strategies are sufficient.

~ classical regions are sufficient for solving such problems


Decidability of timed games

**Theorem [AMPS98,HK99]**

Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and “region-based” strategies are sufficient.

\[\leadsto\] classical regions are sufficient for solving such problems

**Theorem [AM99,BHPR07,JT07]**

Optimal-time reachability timed games are decidable and EXPTIME-complete.

*References*

Back to the example: computing winning states

\[ \begin{align*}
(x \leq 2) & \quad x \geq 1, u_3 \\
& \quad x \leq 1, c_1 \\
x < 1, u_1 & \quad x \geq 2, c_4 \\
x < 1, u_2, x := 0 & \quad x \leq 1, c_3 \\
& \quad c_2 \\
\end{align*} \]
Back to the example: computing winning states

\( \ell_0 \) \( x \leq 2 \) \( x \geq 1, u_3 \)

\( \ell_1 \) \( x \leq 1, c_1 \)

\( \ell_2 \) \( x \geq 2, c_4 \)

\( \ell_3 \) \( c_2 \) \( x \leq 1, c_3 \) 

attraction states

winning states

losing states

\( \ell_0 \)

\( \ell_1 \)

\( \ell_2 \)

\( \ell_3 \)
Back to the example: computing winning states
Back to the example: computing winning states

\begin{verbatim}
\text{\bf \textcolors{red}{} \textcolors{green}{} (x \leq 2) \quad x \geq 1, u_3}
\text{\bf \textcolors{red}{} \textcolors{green}{} x \leq 1, c_1}
\text{\bf \textcolors{green}{} x < 1, u_1}
\text{\bf \textcolors{green}{} x < 1, u_2, x := 0}
\text{\bf \textcolors{red}{} \textcolors{green}{} x \geq 2, c_4}
\text{\bf \textcolors{red}{} \textcolors{green}{} x \leq 1, c_3}
\text{\bf \textcolors{red}{} \textcolors{green}{} c_2}
\end{verbatim}

\text{\bf \textcolors{red}{} \textcolors{green}{} Winning states}

\text{\bf \textcolors{red}{} \textcolors{green}{} Losing states}

\text{\bf \textcolors{green}{} Attrac}
Back to the example: computing winning states

\[
\begin{align*}
\ell_0 & \quad x \leq 2 \\
\ell_1 & \quad x \leq 1, c_1 \quad x > 1, u_3 \\
\ell_2 & \quad x < 1, u_1 \quad x \geq 1, c_2 \quad x \leq 1, c_3 \\
\ell_3 & \quad c_2
\end{align*}
\]
Back to the example: computing winning states
Back to the example: computing winning states
Back to the example: computing winning states

Winning states

Losing states

$\ell_0$

$\ell_1$

$\ell_2$

$\ell_3$
Decidability via attractors
Decidability via attractors

\[ \text{Pred}^a(X) = \{ \bullet \mid \bullet \xrightarrow{a} \bullet \in X \} \]
Decidability via attractors

- \( \text{Pred}^a(X) = \{ \bullet | \bullet \xrightarrow{a} \bullet \in X \} \)

- controllable and uncontrollable discrete predecessors:

\[
\text{cPred}(X) = \bigcup_{a \text{ cont.}} \text{Pred}^a(X) \quad \quad \text{uPred}(X) = \bigcup_{a \text{ uncont.}} \text{Pred}^a(X)
\]
Decidability via attractors

- $\text{Pred}^a(X) = \{ \bullet | \bullet \xrightarrow{a} \bullet \in X \}$

- Controllable and uncontrollable discrete predecessors:

  \[
  \text{cPred}(X) = \bigcup_{a \text{ cont.}} \text{Pred}^a(X) \quad \quad \text{uPred}(X) = \bigcup_{a \text{ uncont.}} \text{Pred}^a(X)
  \]

- Time controllable predecessors:

  $\bullet \xrightarrow{\text{delay} \ t \ t.u.} \bullet \xrightarrow{} \bullet$ should be safe
Decidability via attractors

- \( \text{Pred}^a(X) = \{ \bullet | \bullet \xrightarrow{a} \bullet \in X \} \)

- controllable and uncontrollable discrete predecessors:

\[
\begin{align*}
\text{cPred}(X) &= \bigcup_{a \text{ cont.}} \text{Pred}^a(X) \\
\text{uPred}(X) &= \bigcup_{a \text{ uncont.}} \text{Pred}^a(X)
\end{align*}
\]

- time controllable predecessors:

\[
\begin{align*}
\text{Pred}_\delta(X, \text{Safe}) &= \{ \bullet | \exists t \geq 0, \bullet \xrightarrow{\delta(t)} \bullet \\
\text{and } \forall 0 \leq t' \leq t, \bullet \xrightarrow{\delta(t')} \bullet \in \text{Safe} \}
\end{align*}
\]

\text{should be safe}
Timed games with a reachability objective

We write:

\[ \pi(X) = X \cup \text{Pred}_\delta(\text{cPred}(X), \neg\text{uPred}(\neg X)) \]
Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \text{Pred}_\delta(\text{cPred}(X), \neg \text{uPred}(\neg X))$$

- The states from which one can ensure 😊 in no more than 1 step is:

  $$\text{Attr}_1(😊) = \pi(😊)$$
Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \text{Pred}_\delta(\text{cPred}(X), \neg u\text{Pred}(\neg X))$$

- The states from which one can ensure 😊 in no more than 1 step is:
  $$\text{Attr}_1(😊) = \pi(😊)$$

- The states from which one can ensure 😊 in no more than 2 steps is:
  $$\text{Attr}_2(😊) = \pi(\text{Attr}_1(😊))$$
Timed games with a reachability objective

We write:

\[ \pi(X) = X \cup \text{Pred}_\delta(c\text{Pred}(X), \neg u\text{Pred}(\neg X)) \]

- The states from which one can ensure 😊 in no more than 1 step is:
  \[ \text{Attr}_1(😊) = \pi(😊) \]

- The states from which one can ensure 😊 in no more than 2 steps is:
  \[ \text{Attr}_2(😊) = \pi(\text{Attr}_1(😊)) \]

- ...
Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \text{Pred}_\delta(c\text{Pred}(X), \neg \text{uPred}(\neg X))$$

- The states from which one can ensure 😊 in no more than 1 step is:
  $$\text{Attr}_1(😊) = \pi(😊)$$

- The states from which one can ensure 😊 in no more than 2 steps is:
  $$\text{Attr}_2(😊) = \pi(\text{Attr}_1(😊))$$

- ...  

- The states from which one can ensure 😊 in no more than $n$ steps is:
  $$\text{Attr}_n(😊) = \pi(\text{Attr}_{n-1}(😊))$$
Timed games with a reachability objective

We write:

\[ \pi(X) = X \cup \text{Pred}_\delta(\text{cPred}(X), \neg\text{uPred}(\neg X)) \]

- The states from which one can ensure \( \square \) in no more than 1 step is:
  \[ \text{Attr}_1(\square) = \pi(\square) \]

- The states from which one can ensure \( \square \) in no more than 2 steps is:
  \[ \text{Attr}_2(\square) = \pi(\text{Attr}_1(\square)) \]

- \( \ldots \)

- The states from which one can ensure \( \square \) in no more than \( n \) steps is:
  \[ \text{Attr}_n(\square) = \pi(\text{Attr}_{n-1}(\square)) = \pi^n(\square) \]
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$. 

Does $\pi$ also preserve unions of regions? Yes!

$\text{cPred}(X)$, $\text{uPred}(\neg X)$, $\text{Pred}_\delta(\text{cPred}(X), \neg \text{uPred}(\neg X))$ (but it generates non-convex unions of regions...)

... and is correct.
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does $\pi$ also preserve unions of regions?
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does $\pi$ also preserve unions of regions?

\begin{tikzpicture}
\draw [->] (0,0) -- (5,0);
\draw [->] (0,0) -- (0,5);
\fill[blue!20] (0,0) rectangle (4,4);
\fill[blue!20] (0,4) -- (4,4) -- (4,0) -- (0,0);
\end{tikzpicture}

$c\text{Pred}(X)$
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does $\pi$ also preserve unions of regions?
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $c\text{Pred}(X)$ and $u\text{Pred}(X)$.
- Does $\pi$ also preserve unions of regions?

$\text{Pred}_\delta(c\text{Pred}(X), \neg u\text{Pred}(\neg X))$
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $c\text{Pred}(X)$ and $u\text{Pred}(X)$.

- Does $\pi$ also preserve unions of regions? Yes!

$$\begin{align*}
c\text{Pred}(X) \\
u\text{Pred}(\neg X) \\
\text{Pred}_\delta(c\text{Pred}(X), \neg u\text{Pred}(\neg X))
\end{align*}$$
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $c\text{Pred}(X)$ and $u\text{Pred}(X)$.

- Does $\pi$ also preserve unions of regions? Yes!

(cPred($X$)

uPred($\neg X$)

Pred$_\delta$(cPred($X$), $\neg u\text{Pred}(\neg X)$)

(but it generates non-convex unions of regions...).
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does $\pi$ also preserve unions of regions? Yes!

$\text{cPred}(X) \quad \text{uPred}(\neg X) \quad \text{Pred}_\delta(\text{cPred}(X), \neg \text{uPred}(\neg X))$

(but it generates non-convex unions of regions...)

$\leadsto$ the computation of $\pi^*$(😊) terminates!
Stability w.r.t. regions

- If $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.

- Does $\pi$ also preserve unions of regions? Yes!

(but it generates non-convex unions of regions...)

$\leadsto$ the computation of $\pi^*(\bigcirc)$ terminates!

... and is correct
Timed games with a safety objective

- We can use operator $\tilde{\pi}$ defined by

$$\tilde{\pi}(X) = \text{Pred}_\delta(X \cap \text{cPred}(X), \neg \text{uPred}(\neg X))$$

instead of $\pi$, and compute $\tilde{\pi}^*(\neg \Box)$
Timed games with a safety objective

- We can use operator $\tilde{\pi}$ defined by

$$\tilde{\pi}(X) = \text{Pred}_\delta(X \cap c\text{Pred}(X), \neg u\text{Pred}(\neg X))$$

instead of $\pi$, and compute $\tilde{\pi}^*(\neg \exists)$.

- It is also stable w.r.t. regions.
Outline

1. Introduction

2. Overview of “old” results
   - Weighted timed automata
   - Timed games
   - Weighted timed games

3. Some recent developments
   - Undecidability of the value problem
   - Approximation of the optimal cost
   - Back to the undecidability

4. Conclusion
A simple timed game

\[ x \leq 2, c, y := 0 \]

\[(y = 0)\]

\[ x = 2, c \]

\[ x = 2, c \]

\[ \inf_{0 \leq t \leq 2} \max (5t + 10(2 - t) + 1, 5t + (2 - t) + 7) = 14 + \frac{1}{3}; \]

strategy: wait in \( \ell_0 \), and when \( t = \frac{4}{3} \), go to \( \ell_1 \)
A simple weighted timed game

\[ x \leq 2, c, y := 0 \]

\[ (y = 0) \]

\[ x = 2, c \]

\[ +1 \]

\[ +10 \]

\[ +1 \]
A simple weighted timed game

Question: what is the optimal cost we can ensure while reaching 😊?
A simple weighted timed game

\[ +5 \quad x \leq 2, c, y := 0 \quad (y = 0) \quad +10 \quad +1 \quad x = 2, c \]

**Question:** what is the optimal cost we can ensure while reaching 😊?

\[ 5t + 10(2 - t) + 1 \]
A simple weighted timed game

Question: what is the optimal cost we can ensure while reaching 😊?

\[ 5t + 10(2 - t) + 1, \quad 5t + (2 - t) + 7 \]
A simple weighted timed game

\[ \ell_0 \xrightarrow{+5} \ell_1 \xrightarrow{+10} \ell_2 \xrightarrow{+1} \text{smiley} \] 
\[ \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{+1} \ell_2 \]

**Question:** what is the optimal cost we can ensure while reaching \text{smiley}?

\[
\max \left( 5t + 10(2 - t) + 1, 5t + (2 - t) + 7 \right)
\]
A simple weighted timed game

Question: what is the optimal cost we can ensure while reaching ☺︎?

$$\inf_{0 \leq t \leq 2} \max \left( 5t + 10(2 - t) + 1, \ 5t + (2 - t) + 7 \right) = 14 + \frac{1}{3}$$
A simple weighted timed game

Question: what is the optimal cost we can ensure while reaching 😊?

\[
\inf_{0 \leq t \leq 2} \max \left( 5t + 10(2 - t) + 1 , \ 5t + (2 - t) + 7 \right) = 14 + \frac{1}{3}
\]

\sim strategy: wait in \ell_0, and when \( t = \frac{4}{3} \), go to \ell_1
Optimal reachability in weighted timed games (1)

This topic has been fairly hot these last fifteen years...

[LMM02,ABM04,BCFL04,BBR05,BBM06,BLMR06,Rut11,HIM13,BGK+14]

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata (TCS@02).
[BBR05] Brihaye, Bruyère, Raskin. On optimal timed strategies (FORMATS’05).
[BBM06] Bouyer, Brihaye, Markey. Improved undecidability results on weighted timed automata (Information Processing Letters).
[BGK+14] Brihaye, Geeraerts, Krishna, Manasa, Monmege, Trivedi. Adding Negative Prices to Priced Timed Games (CONCUR’14).
Optimal reachability in weighted timed games (1)

This topic has been fairly hot these last fifteen years...

[LMM02, ABM04, BCFL04, BBR05, BBM06, BLMR06, Rut11, HIM13, BGK+14]

[LMM02]
Tree-like weighted timed games can be solved in 2EXPTIME.
Optimal reachability in weighted timed games (1)

This topic has been fairly hot these last fifteen years...

[LMM02,ABM04,BCFL04,BBR05,BBM06,BLMR06,Rut11,HIM13,BGK+14]

[LMM02]
Tree-like weighted timed games can be solved in 2EXPTIME.

[ABM04,BCFL04]
Depth-\(k\) weighted timed games can be solved in EXPTIME. There is a symbolic algorithm to solve weighted timed games with a strongly non-Zeno cost.
Optimal reachability in weighted timed games (2)

[BBR05, BBM06]

In weighted timed games, the optimal cost cannot be computed, as soon as games have three clocks or more.
Optimal reachability in weighted timed games (2)

[BBR05, BBM06]

In weighted timed games, the optimal cost cannot be computed, as soon as games have three clocks or more.

[BLMR06, Rut11, HIM13, BGK+14]

Turn-based optimal timed games are decidable in EXPTIME (resp. PTIME) when automata have a single clock (resp. with two rates). They are PTIME-hard.
What is easier with a single clock?

- Memoryless strategies can be non-optimal...

\[ \ell_0 \xrightarrow{+2} \ell_1 \xrightarrow{x=1} \text{happy face} \]

- Memoryless almost-optimal strategies will be sufficient.
What is easier with a single clock?

- Memoryless strategies can be non-optimal...

... but memoryless almost-optimal strategies will be sufficient.
What is easier with a single clock?

- Memoryless strategies can be non-optimal...

\[
\begin{align*}
\ell_0 &\xrightarrow{+2} \ell_1 \\
(x \leq 1) \quad &\text{\textcolor{orange}{x=1}} \\
&\xrightarrow{x < 1} \ell_1 \\
&\xrightarrow{x := 0} \ell_1 \\
&\xrightarrow{x > 0} \ell_0
\end{align*}
\]

... but memoryless almost-optimal strategies will be sufficient.

- Key: resetting the clock somehow resets the history...
What is easier with a single clock?

- Memoryless strategies can be non-optimal...

  ![Diagram](image)

  ... but memoryless almost-optimal strategies will be sufficient.

- Key: resetting the clock somehow resets the history...

- By unfolding and removing one by one the locations, we can synthesize **memoryless almost-optimal** winning strategies.
What is easier with a single clock?

- Memoryless strategies can be non-optimal...

  \[(x \leq 1)\]

  \[\ell_0 \xrightarrow{+2} \ell_1 \xrightarrow{x=1} \]  \[x<1\]

  \[x:=0\]

  \[x>0\]

  \[x=1\]

  ... but memoryless almost-optimal strategies will be sufficient.

- Key: resetting the clock somehow resets the history...

- By unfolding and removing one by one the locations, we can synthesize memoryless almost-optimal winning strategies.

- Rather involved proofs of correctness
Introduction  Overview of "old" results  Some recent developments  Conclusion
Weighted timed automata  Timed games  Weighted timed games

\[ \sigma(c_2, x) = \begin{cases} 
    c_2^{out} & \text{if } 0 \leq x < 2/5 \\
    c_2 & \text{if } 2/5 \leq x < 1/2 \\
    u_2 & \text{if } 1/2 \leq x \leq 1 
\end{cases} \]
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$. 
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

The cost is increased by $x_0$

The cost is increased by $1 - x_0$
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.  

In $\mathcal{K}$, cost $= 2x_0 + (1 - y_0) + 2$ if $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$ if $y_0 > 2x_0$, player 2 chooses the second branch: cost $> 3$ if $y_0 = 2x_0$, in both branches, cost $= 3$; player 2 can enforce cost $3 + |y_0 - 2x_0|$.

Player 1 has a winning strategy with cost $\leq 3$ iff $y_0 = 2x_0$. 

\begin{align*}
x &= x_0 \\
y &= y_0 \\
z &= 0
\end{align*}
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In $\mathbb{Z}_+$, cost $= 2x_0 + (1 - y_0) + 2$
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

\[
\begin{align*}
\text{In } \boxed{\text{ }} \text{, cost } &= 2x_0 + (1 - y_0) + 2 \\
\text{In } \boxed{\text{ }} \text{, cost } &= 2(1 - x_0) + y_0 + 1
\end{align*}
\]
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In $\bigcirc$, cost = $2x_0 + (1 - y_0) + 2$
- In $\square$, cost = $2(1 - x_0) + y_0 + 1$
- if $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In $\smiley$, cost $= 2x_0 + (1 - y_0) + 2$
- In $\frowny$, cost $= 2(1 - x_0) + y_0 + 1$

If $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$
If $y_0 > 2x_0$, player 2 chooses the second branch: cost $> 3$
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In the green box, cost $= 2x_0 + (1 - y_0) + 2$
- In the pink box, cost $= 2(1 - x_0) + y_0 + 1$

- If $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$
- If $y_0 > 2x_0$, player 2 chooses the second branch: cost $> 3$
- If $y_0 = 2x_0$, in both branches, cost $= 3$
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In $\ddot{\smile}$, cost $= 2x_0 + (1 - y_0) + 2$
- In $\ddot{\frown}$, cost $= 2(1 - x_0) + y_0 + 1$

- if $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$
- if $y_0 > 2x_0$, player 2 chooses the second branch: cost $> 3$
- if $y_0 = 2x_0$, in both branches, cost $= 3$

$\implies$ player 2 can enforce cost $3 + |y_0 - 2x_0|$
Computing the optimal cost: why is that hard?

Given two clocks $x$ and $y$, we can check whether $y = 2x$.

- In $\smiley$, cost $= 2x_0 + (1 - y_0) + 2$
- In $\frown$, cost $= 2(1 - x_0) + y_0 + 1$

- If $y_0 < 2x_0$, player 2 chooses the first branch: cost $> 3$
  - If $y_0 > 2x_0$, player 2 chooses the second branch: cost $> 3$
  - If $y_0 = 2x_0$, in both branches, cost $= 3$

  $\implies$ player 2 can enforce cost $3 + |y_0 - 2x_0|$

- Player 1 has a winning strategy with cost $\leq 3$ iff $y_0 = 2x_0$
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:
- each instruction is encoded as a module;
- the counter values $c_1$ and $c_2$ are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:
- each instruction is encoded as a module;
- the counter values $c_1$ and $c_2$ are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:
  - each instruction is encoded as a module;
  - the counter values $c_1$ and $c_2$ are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.

Globally, $(x \leq 1, y \leq 1, u \leq 1)$

$$x=1, x:=0 \quad \lor \quad y=1, y:=0$$

Test $y(x=2z)$
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:

- each instruction is encoded as a module;
- the counter values $c_1$ and $c_2$ are encoded by two clocks:

\[ x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}} \]

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:
- each instruction is encoded as a module;
- the counter values $c_1$ and $c_2$ are encoded by two clocks:

$$x = \frac{1}{2^{c_1}} \quad \text{and} \quad y = \frac{1}{2^{c_2}}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:
- each instruction is encoded as a module;
- the counter values $c_1$ and $c_2$ are encoded by two clocks:

$$x = \frac{1}{2c_1} \quad \text{and} \quad y = \frac{1}{2c_2}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:
- each instruction is encoded as a module;
- the counter values $c_1$ and $c_2$ are encoded by two clocks:

$$x = \frac{1}{2c_1} \quad \text{and} \quad y = \frac{1}{2c_2}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.
Computing the optimal cost: why is that hard?

Player 1 will simulate a two-counter machine:
- each instruction is encoded as a module;
- the counter values $c_1$ and $c_2$ are encoded by two clocks:

$$x = \frac{1}{2c_1} \quad \text{and} \quad y = \frac{1}{2c_2}$$

The two-counter machine has a halting computation iff player 1 has a winning strategy to ensure a cost no more than 3.
Shape of the reduction
Shape of the reduction
Are we done?
Outline

1. Introduction

2. Overview of “old” results
   - Weighted timed automata
   - Timed games
   - Weighted timed games

3. Some recent developments
   - Undecidability of the value problem
   - Approximation of the optimal cost
   - Back to the undecidability

4. Conclusion
Are we done?
Are we done?  No! Let’s be a bit more precise!
Are we done? No! Let’s be a bit more precise!

Given $\mathcal{G}$ a weighted timed game,

- a strategy $\sigma$ is winning whenever all its outcomes are winning;
Are we done? No! Let’s be a bit more precise!

Given $G$ a weighted timed game,

- a strategy $\sigma$ is winning whenever all its outcomes are winning;
- Cost of a winning strategy $\sigma$:

$$\text{cost}(\sigma) = \sup\{\text{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target}\}$$
Are we done? No! Let’s be a bit more precise!

Given $\mathcal{G}$ a weighted timed game,
- a strategy $\sigma$ is winning whenever all its outcomes are winning;
- Cost of a winning strategy $\sigma$:
  $$\text{cost}(\sigma) = \sup\{\text{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target}\}$$
- Optimal cost:
  $$\text{optcost}_\mathcal{G} = \inf_{\sigma \text{ winning strat.}} \text{cost}(\sigma)$$
  (set it to $+\infty$ if there is no winning strategy)
Are we done? No! Let’s be a bit more precise!

Given \( G \) a weighted timed game,
- a strategy \( \sigma \) is winning whenever all its outcomes are winning;
- Cost of a winning strategy \( \sigma \):
  \[
  \text{cost}(\sigma) = \sup \{ \text{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target} \}
  \]
- Optimal cost:
  \[
  \text{optcost}_G = \inf_{\sigma \text{ winning strat.}} \text{cost}(\sigma)
  \]
  (set it to \( +\infty \) if there is no winning strategy)

Two problems of interest
- The value problem asks, given \( G \) and a threshold \( \triangleright c \), whether \( \text{optcost}_G \triangleright c \)?
Are we done? No! Let’s be a bit more precise!

Given $G$ a weighted timed game,

- a strategy $\sigma$ is winning whenever all its outcomes are winning;
- Cost of a winning strategy $\sigma$:

$$\text{cost}(\sigma) = \sup\{\text{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target}\}$$

- Optimal cost:

$$\text{optcost}_G = \inf_{\sigma \text{ winning strat.}} \text{cost}(\sigma)$$

(set it to $+\infty$ if there is no winning strategy)

Two problems of interest

- The value problem asks, given $G$ and a threshold $\bowtie c$, whether $\text{optcost}_G \bowtie c$?
- The existence problem asks, given $G$ and a threshold $\bowtie c$, whether there exists a winning strategy in $G$ such that $\text{cost}(\sigma) \bowtie c$?
Are we done? No! Let’s be a bit more precise!

Given $\mathcal{G}$ a weighted timed game,

- a strategy $\sigma$ is winning whenever all its outcomes are winning;
- Cost of a winning strategy $\sigma$:
  \[
  \text{cost}(\sigma) = \sup \{ \text{cost}(\rho) \mid \rho \text{ outcome of } \sigma \text{ up to the target} \}
  \]

- Optimal cost:
  \[
  \text{optcost}_{\mathcal{G}} = \inf_{\sigma \text{ winning strat.}} \text{cost}(\sigma)
  \]
  (set it to $+\infty$ if there is no winning strategy)

Two problems of interest

- The value problem asks, given $\mathcal{G}$ and a threshold $\triangleright c$, whether $\text{optcost}_{\mathcal{G}} \triangleright c$?
- The existence problem asks, given $\mathcal{G}$ and a threshold $\triangleleft c$, whether there exists a winning strategy in $\mathcal{G}$ such that $\text{cost}(\sigma) \triangleleft c$?

Note: These problems are distinct...
The value of the game is 3, but no strategy has cost 3.
The value of the game is 3, but no strategy has cost 3.
The value of the game is 3, but no strategy has cost 3.
Weighted timed automata

In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.
Weighted timed automata

In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE. The value problem is PSPACE-complete in weighted timed automata. Almost-optimal winning schedules can be computed.
Weighted timed automata

In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE. The value problem is PSPACE-complete in weighted timed automata. Almost-optimal winning schedules can be computed.

Weighted timed games

Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.
- Weighted timed automata

In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE. The value problem is PSPACE-complete in weighted timed automata. Almost-optimal winning schedules can be computed.

- Weighted timed games

Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock. The value problem is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.
Weighted timed automata

In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.
The value problem is PSPACE-complete in weighted timed automata. Almost-optimal winning schedules can be computed.

Weighted timed games

Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.
The value problem is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.

There is a symbolic algorithm to solve weighted timed games with a strongly non-Zeno cost.
**Weighted timed automata**

In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.

The *value problem* is PSPACE-complete in weighted timed automata. Almost-optimal winning schedules can be computed.

**Weighted timed games**

Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.

The *value problem* is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.

There is a symbolic algorithm to solve weighted timed games with a strongly non-Zeno cost.

The *value problem* can be decided in EXPTIME in weighted timed games with a strongly non-Zeno cost. Almost-optimal winning strategies can be computed.
Weighted timed automata

In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE. The value problem is PSPACE-complete in weighted timed automata. Almost-optimal winning schedules can be computed.

Weighted timed games

Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock. The value problem is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.

There is a symbolic algorithm to solve weighted timed games with a strongly non-Zeno cost. The value problem can be decided in EXPTIME in weighted timed games with a strongly non-Zeno cost. Almost-optimal winning strategies can be computed.

In weighted timed games, the optimal cost cannot be computed, as soon as games have three clocks or more.
-weighted timed automata

In weighted timed automata, the optimal cost is an integer, and can be computed in PSPACE.
The value problem is PSPACE-complete in weighted timed automata. Almost-optimal winning schedules can be computed.

-weighted timed games

Turn-based optimal timed games are decidable in EXPTIME when automata have a single clock.
The value problem is decidable in EXPTIME in single-clock weighted timed games. Almost-optimal memoryless winning strategies can be computed.

There is a symbolic algorithm to solve weighted timed games with a strongly non-Zeno cost.
The value problem can be decided in EXPTIME in weighted timed games with a strongly non-Zeno cost. Almost-optimal winning strategies can be computed.

In weighted timed games, the optimal cost cannot be computed, as soon as games have three clocks or more.
The existence problem is undecidable in weighted timed games.
Outline of the rest of the talk

1. Show that the **value problem** is undecidable in weighted timed games
Outline of the rest of the talk

1. Show that the value problem is undecidable in weighted timed games
   - This is intellectually satisfactory to not have this discrepancy in the set of results
Outline of the rest of the talk

Show that the value problem is undecidable in weighted timed games

- This is intellectually satisfactory to not have this discrepancy in the set of results
- A first proof based on a diagonal construction (originally proposed in the context of quantitative temporal logics [BMM14])

Outline of the rest of the talk

1. Show that the **value problem** is undecidable in weighted timed games
   - This is intellectually satisfactory to not have this discrepancy in the set of results
   - A first proof based on a diagonal construction (originally proposed in the context of quantitative temporal logics [BMM14])
   - A second direct proof
Outline of the rest of the talk

1. Show that the value problem is undecidable in weighted timed games
   - This is intellectually satisfactory to not have this discrepancy in the set of results
   - A first proof based on a diagonal construction (originally proposed in the context of quantitative temporal logics [BMM14])
   - A second direct proof

2. Propose an approximation algorithm for a large class of weighted timed games (that comprises the class of games used for proving the above undecidability)
   - Almost-optimality in practice should be sufficient
   - Even when we know how to compute the value, we are only able to synthesize almost-optimal strategies...
Outline

1 Introduction

2 Overview of “old” results
   - Weighted timed automata
   - Timed games
   - Weighted timed games

3 Some recent developments
   - Undecidability of the value problem
   - Approximation of the optimal cost
   - Back to the undecidability

4 Conclusion
A snapshot on the undecidability proof

Instruction

Test module
A snapshot on the undecidability proof

Instruction

Test module
A snapshot on the undecidability proof

Leave with cost $3 + \frac{1}{2^n}$ ($n$: length of the path)
A snapshot on the undecidability proof

\[ \mathcal{M} \text{ does not halt iff the value of } G_\mathcal{M} \text{ is } 3 \]

Leave with cost \( 3 + \frac{1}{2^n} \) \((n: \text{ length of the path})\)
**Theorem [BJM15]**

The value problem is undecidable in weighted timed games (with four clocks or more).

- Remark on the reduction:
  - Cost 0 within the core of the game
  - The rest of the game is acyclic
Outline

1 Introduction

2 Overview of “old” results
   - Weighted timed automata
   - Timed games
   - Weighted timed games

3 Some recent developments
   - Undecidability of the value problem
   - Approximation of the optimal cost
   - Back to the undecidability

4 Conclusion
Optimal cost is computable...

... when cost is strongly non-zeno. \[\text{[AM04, BCFL04]}\]

That is, there exists $\kappa > 0$ such that for every region cycle $C$, for every real run $\varrho$ read on $C$,

$$\text{cost}(\varrho) \geq \kappa$$

Optimal cost is not computable...

... when cost is almost-strongly non-zeno. \[\text{[BJM15]}\]

That is, there exists $\kappa > 0$ such that for every region cycle $C$, for every real run $\varrho$ read on $C$,

$$\text{cost}(\varrho) \geq \kappa \quad \text{or} \quad \text{cost}(\varrho) = 0$$

Note: In both cases, we can assume $\kappa = 1$. 

Optimal cost is computable...

... when cost is strongly non-zeno.  

That is, there exists $\kappa > 0$ such that for every region cycle $C$, for every real run $\varrho$ read on $C$,

$$\text{cost}(\varrho) \geq \kappa$$

[AM04,BCFL04]

Optimal cost is not computable... but is approximable!

... when cost is almost-strongly non-zeno.  

That is, there exists $\kappa > 0$ such that for every region cycle $C$, for every real run $\varrho$ read on $C$,

$$\text{cost}(\varrho) \geq \kappa \quad \text{or} \quad \text{cost}(\varrho) = 0$$

[BJM15]

Note: In both cases, we can assume $\kappa = 1$. 

Approximation of the optimal cost

Theorem

Let $G$ be a weighted timed game, in which the cost is almost-strongly non-zeno. For every $\epsilon > 0$, one can compute:

- two values $v_\epsilon^-$ and $v_\epsilon^+$ such that

\[ |v_\epsilon^+ - v_\epsilon^-| < \epsilon \quad \text{and} \quad v_\epsilon^- \leq \text{optcost}_G \leq v_\epsilon^+ \]
Approximation of the optimal cost

**Theorem**

Let $\mathcal{G}$ be a weighted timed game, in which the cost is almost-strongly non-zeno. For every $\epsilon > 0$, one can compute:

- two values $v_\epsilon^-$ and $v_\epsilon^+$ such that

  $$|v_\epsilon^+ - v_\epsilon^-| < \epsilon \quad \text{and} \quad v_\epsilon^- \leq \text{optcost}_G \leq v_\epsilon^+$$

- one strategy $\sigma_\epsilon$ such that

  $$\text{optcost}_G \leq \text{cost}(\sigma_\epsilon) \leq \text{optcost}_G + \epsilon$$

[It is an $\epsilon$-optimal winning strategy]
Approximation of the optimal cost

Theorem

Let $G$ be a weighted timed game, in which the cost is almost-strongly non-zeno. For every $\epsilon > 0$, one can compute:

- two values $v_{\epsilon}^-$ and $v_{\epsilon}^+$ such that
  \[ |v_{\epsilon}^+ - v_{\epsilon}^-| < \epsilon \quad \text{and} \quad v_{\epsilon}^- \leq \text{optcost}_G \leq v_{\epsilon}^+ \]

- one strategy $\sigma_\epsilon$ such that
  \[ \text{optcost}_G \leq \text{cost}(\sigma_\epsilon) \leq \text{optcost}_G + \epsilon \]

[It is an $\epsilon$-optimal winning strategy]

- Standard technics: unfold the game to get more precision, and compute two adjacency sequences
Approximation of the optimal cost

**Theorem**

Let $\mathcal{G}$ be a weighted timed game, in which the cost is almost-strongly non-zeno. For every $\epsilon > 0$, one can compute:

- two values $v_\epsilon^-$ and $v_\epsilon^+$ such that
  $$|v_\epsilon^+ - v_\epsilon^-| < \epsilon \quad \text{and} \quad v_\epsilon^- \leq \text{optcost}_G \leq v_\epsilon^+$$
- one strategy $\sigma_\epsilon$ such that
  $$\text{optcost}_G \leq \text{cost}(\sigma_\epsilon) \leq \text{optcost}_G + \epsilon$$

[it is an $\epsilon$-optimal winning strategy]

- Standard technics: unfold the game to get more precision, and compute two adjacency sequences
- This is not possible here
  There might be runs with prefixes of arbitrary length and cost 0 (e.g. the game of the undecidability proof)
Idea for approximation

Idea

Only partially unfold the game:
- Keep components with cost 0 untouched – we call it the kernel
- Unfold the rest of the game
Idea for approximation

Idea

Only partially unfold the game:
- Keep components with cost 0 untouched – we call it the kernel
- Unfold the rest of the game

First: split the game along regions!
Semi-unfolding

Hypothesis: \( \text{cost} > 0 \) implies \( \text{cost} \geq \kappa \)

Conclusion: we can stop unfolding the game after \( N \) steps (e.g. \( N = (M + 2) \cdot |R(A)| \), where \( M \) is a pre-computed bound on \( \text{optcost} \)).
Semi-unfolding

Hypothesis: \( \text{cost} > 0 \) implies \( \text{cost} \geq \kappa \)

Conclusion: we can stop unfolding the game after \( N \) steps (e.g. \( N = (M + 2) \cdot |R(A)| \), where \( M \) is a pre-computed bound on \( \text{optcost} \)).

Undecidability of the value problem Approximation of the optimal cost Back to the undecidability
Semi-unfolding

Undecidability of the value problem
Approximation of the optimal cost
Back to the undecidability
Semi-unfolding

Hypothesis:
\[ \text{cost} > 0 \implies \text{cost} \geq \kappa \]

K

Only cost 0

Kernel \( \mathcal{K} \)

(\( \ell, r \))
Semi-unfolding

Hypothesis:
\[ \text{cost} > 0 \implies \text{cost} \geq \kappa \]

Conclusion: we can stop unfolding the game after \( N \) steps
\( (\text{e.g. } N = (M + 2) \cdot |\mathcal{R}(A)|, \text{ where } M \text{ is a pre-computed bound on } \text{optcost}_G) \)
Approximation scheme
Approximation scheme
Approximation scheme
Approximation scheme
Approximation scheme
First step: Tree-like parts

\[ \text{Goes back to [LMM02]} \]
First step: Tree-like parts

\[ O(\ell, v) = \inf_{t' \mid v + t' = g'} \max(\alpha), (\beta) \]
\[ (\alpha) = t' c + c' + O(\ell', v') \]
\[ (\beta) = \sup_{t'' \leq t' \mid v + t'' = g''} t' c + c'' + O(\ell'', v'') \]

\[ \ell', c', g', Y' \]
\[ \ell'', c'', g'', Y'' \]

\[ \leadsto \text{Goes back to [LMM02]} \]

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata (TCS@02).
First step: Tree-like parts

\[ O(\ell, v) = \]

\[ g', Y', c', o(\ell', v') \]
\[ g'', Y'', c'', o(\ell'', v'') \]

\[ \sim \text{ Goes back to [LMM02]} \]

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata (TCS@02).
First step: Tree-like parts

\[ O(\ell, v) = \inf_{t' \mid v + t' = g'} t' \]

\[ O(\ell', v') \]

\[ O(\ell'', v'') \]

Goes back to [LMM02]

La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata (TCS@02).
First step: Tree-like parts

$O(\ell, v) = \inf_{t'} \max\left( \left. v + t' \right| = g' \right)$

\[\sim \text{Goes back to [LMM02]}\]

[156x266] Introduction Overview of “old” results Some recent developments Conclusion
Undecidability of the value problem Approximation of the optimal cost Back to the undecidability

First step: Tree-like parts

$O(\ell', v')$ $O(\ell'', v'')$

$O(\ell, v) = \inf_{t'} \max\left( \left. v + t' \right| = g' \right)$

\[\sim \text{Goes back to [LMM02]}\]

[156x266] Introduction Overview of “old” results Some recent developments Conclusion
Undecidability of the value problem Approximation of the optimal cost Back to the undecidability

First step: Tree-like parts

$O(\ell', v')$ $O(\ell'', v'')$

$O(\ell, v) = \inf_{t'} \max\left( \left. v + t' \right| = g' \right)$

\[\sim \text{Goes back to [LMM02]}\]

[156x266] Introduction Overview of “old” results Some recent developments Conclusion
Undecidability of the value problem Approximation of the optimal cost Back to the undecidability

First step: Tree-like parts

$O(\ell', v')$ $O(\ell'', v'')$

$O(\ell, v) = \inf_{t'} \max\left( \left. v + t' \right| = g' \right)$

\[\sim \text{Goes back to [LMM02]}\]

[156x266] Introduction Overview of “old” results Some recent developments Conclusion
Undecidability of the value problem Approximation of the optimal cost Back to the undecidability

First step: Tree-like parts

$O(\ell', v')$ $O(\ell'', v'')$

$O(\ell, v) = \inf_{t'} \max\left( \left. v + t' \right| = g' \right)$

\[\sim \text{Goes back to [LMM02]}\]
First step: Tree-like parts

\[ O(\ell, v) = \inf_{t' \mid v + t' = g'} \max \left( (\alpha), \left( (\beta) = \sup_{t'' \leq t' \mid v + t'' = g''} t'' c + c' + O(\ell', v') \right) \right) \]

\[ v' = \Delta(Y' \leftarrow 0)(v + t') \]

\[ [LMM02] \text{La Torre, Mukhopadhyay, Murano. Optimal-reachability and control for acyclic weighted timed automata (TCS@02).} \]
First step: Tree-like parts

\[ O(\ell, v) = \inf_{t' | v + t' = g'} \max((\alpha), (\beta)) \]

\[ (\alpha) = t'c' + c' + O(\ell', v') \]

\[ (\beta) = \sup_{t'' \leq t' | v + t'' = g''} t''c + c'' + O(\ell'', v'') \]

\[ v' = [Y' \leftarrow 0](v + t') \]

\[ v'' = [Y'' \leftarrow 0](v + t'') \]

\[ O(\ell', v') \]

\[ O(\ell'', v'') \]

\[ g', Y' \]

\[ g'', Y'' \]

\[ c' \]

\[ c'' \]

\[ \ell' \]

\[ \ell'' \]

\[ c \]

\[ \ell \]

\[ g', Y' \]

\[ g'', Y'' \]

\[ c' \]

\[ c'' \]

\[ \ell' \]

\[ \ell'' \]

\[ c \]

\[ \ell \]

\[ \sim \text{ Goes back to} \ [\text{LMM02}] \]
Second step: Kernels

Output cost functions $f$
Second step: Kernels

1. Refine the regions such that $f$ differs of at most $\epsilon$ within a small region.

Output cost functions $f$
Second step: Kernels

1. Refine the regions such that $f$ differs of at most $\epsilon$ within a small region.

Output cost functions $f$
Second step: Kernels

1. Refine the regions such that $f$ differs of at most $\epsilon$ within a small region

Output cost functions $f$
Second step: Kernels

1. Refine the regions such that \( f \) differs of at most \( \epsilon \) within a small region

2. Under- and over-approximate by piecewise constant functions \( f_{\epsilon}^- \) and \( f_{\epsilon}^+ \)
Second step: Kernels

Refine/split the kernel along the new small regions and fix $f_\epsilon^-$ or $f_\epsilon^+$, write $f_\epsilon$

$\epsilon$:

- $f_\epsilon$: constant
- $f_\epsilon$: constant
Second step: Kernels

3. Refine/split the kernel along the new small regions and fix $f_\epsilon^-$ or $f_\epsilon^+$, write $f_\epsilon$

4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by $f_\epsilon$)
Second step: Kernels

Refine/split the kernel along the new small regions and fix $f_\epsilon^-$ or $f_\epsilon^+$, write $f_\epsilon$

Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by $f_\epsilon$)

Those can be solved using standard technics based on attractors: small regions are sufficient, and the local optimal cost (for output $f_\epsilon$) is constant within a small region

$f_\epsilon$: constant  \quad  f_\epsilon$: constant
Second step: Kernels

- Refine/split the kernel along the new small regions and fix $f^-_\epsilon$ or $f^+_\epsilon$, write $f_\epsilon$

- Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by $f_\epsilon$)

- Those can be solved using standard technics based on attractors: small regions are sufficient, and the local optimal cost (for output $f_\epsilon$) is constant within a small region
Second step: Kernels

3. Refine/split the kernel along the new small regions and fix $f_{\epsilon^-}$ or $f_{\epsilon^+}$, write $f_\epsilon$.

4. Since cost is 0 everywhere, the resulting game is nothing more than a reachability timed game with an order on target (output) edges (given by $f_\epsilon$).

5. Those can be solved using standard technics based on attractors: small regions are sufficient, and the local optimal cost (for output $f_\epsilon$) is constant within a small region.

~ We have computed $\epsilon$-approximations of the optimal cost, which are constant within small regions. Corresponding strategies can be inferred.
Outline

1 Introduction

2 Overview of “old” results
   - Weighted timed automata
   - Timed games
   - Weighted timed games

3 Some recent developments
   - Undecidability of the value problem
   - Approximation of the optimal cost
   - Back to the undecidability

4 Conclusion
Consequence of the approximation algorithm

**Theorem**

The value problem is co-recursively enumerable (for almost-strongly non-zeno weighted timed games), but not recursively enumerable.
Outline

1. Introduction

2. Overview of “old” results
   - Weighted timed automata
   - Timed games
   - Weighted timed games

3. Some recent developments
   - Undecidability of the value problem
   - Approximation of the optimal cost
   - Back to the undecidability

4. Conclusion
Conclusion

Summary of the talk

- Quick overview of results concerning the optimal reachability problem in weighted timed games
- New insight into the value problem for this model:
  - Undecidability of this problem
  - Approximability of the optimal cost (under some conditions)
Conclusion

### Summary of the talk

- Quick overview of results concerning the optimal reachability problem in weighted timed games
- New insight into the value problem for this model:
  - Undecidability of this problem
  - Approximability of the optimal cost (under some conditions)

### Future work

- Improve the approximation scheme \((2\text{EXP}(|G|) \cdot \left(1/\varepsilon\right)^{|X|})\)
- Extend to the whole class of weighted timed games? understand why it is not possible
- Assume stochastic uncertainty
Summary of the talk

- Quick overview of results concerning the optimal reachability problem in weighted timed games
- New insight into the value problem for this model:
  - Undecidability of this problem
  - Approximability of the optimal cost (under some conditions)

Future work

- Improve the approximation scheme \(2\text{EXP}(|G|) \cdot \left(\frac{1}{\epsilon}\right)^{|X|}\)
- Extend to the whole class of weighted timed games? understand why it is not possible
- Assume stochastic uncertainty
- Is the value of any game a rational number?
Conclusion

Summary of the talk

- Quick overview of results concerning the optimal reachability problem in weighted timed games
- New insight into the value problem for this model:
  - Undecidability of this problem
  - Approximability of the optimal cost (under some conditions)

Future work

- Improve the approximation scheme \( (2\text{EXP}(|G|) \cdot \left(1/\epsilon \right)^{|X|}) \)
- Extend to the whole class of weighted timed games? understand why it is not possible
- Assume stochastic uncertainty
- Is the value of any game a rational number?
- Understand the multiplayer setting (see next slides)
Nash equilibria in weighted timed games

The setting

- One weight function per player, one target state
- Payoff\(_i\): weight\(_i\) of the outcome if the target is reached; \(+\infty\) otherwise (note: the smaller, the better)
- Nash equilibrium: a strategy profile such that the payoff of each player cannot be improved by unilateral deviation by that player
Nash equilibria in weighted timed games

The setting

- One weight function per player, one target state
- **Payoff** \( i \): weight \( i \) of the outcome if the target is reached; \(+\infty\) otherwise (note: the smaller, the better)
- **Nash equilibrium**: a strategy profile such that the payoff of each player cannot be improved by unilateral deviation by that player

Theorem

In a two-player (non-zero-sum) weighted timed game as given above, we cannot decide whether there is a Nash equilibrium.

\[ \leadsto \text{ inspired by a result in Romain Brenguier’s Master thesis} \]

(originally one clock, and negative/positive weights)
An interesting gadget with no Nash equilibrium

In this game, if there is a NE, then the payoff of each player is no more than 3.
Add⁺(x)

The cost is increased by $x_0$

Add⁻(x)

The cost is increased by $1 - x_0$

Two possibilities:

Player 2:

Player 1: $3 + (y_0 - 2x_0)$

Player 2: $3 - (y_0 - 2x_0) + \epsilon$

Player 2 has a strategy to get payoff $3 - |y_0 - 2x_0| + \epsilon$ (with $\epsilon > 0$) and give payoff $3 + |y_0 - 2x_0|$ to Player 1.

There is a Nash Equilibrium if and only if the two-counter machine halts.
Add\(^+(x)\)

\[
\begin{align*}
y &= 1, y := 0 \\
x &= 1, x := 0 \\
z &= 0
\end{align*}
\]

Add\(^-(x)\)

\[
\begin{align*}
y &= 1, y := 0 \\
x &= 1, x := 0 \\
z &= 0
\end{align*}
\]

**cost\(_1\)** is increased by \(x_0\)

**cost\(_2\)** is increased by \(1 - x_0\)

---

Two possibilities:

**Player 2:**

\[
\begin{align*}
&\text{Player 1: } 3 + (y_0 - 2x_0) \\
&\quad - (y_0 - 2x_0) + \epsilon \\
&\text{Player 2: } 3 - (y_0 - 2x_0) + \epsilon
\end{align*}
\]

Player 2 has a strategy to get payoff 3

\[-|y_0 - 2x_0| + \epsilon\]

and give payoff 3

\[+|y_0 - 2x_0| + \epsilon\]

There is a NE if and only if the two-counter machine halts.
Introduction
Overview of "old" results
Some recent developments
Conclusion

Two possibilities:  
Player 2:

Player 1:  \[ 3 + (y_0 - 2x_0) \quad 3 - (y_0 - 2x_0) \]

Player 2:  \[ 3 - (y_0 - 2x_0) + \epsilon \quad 3 + (y_0 - 2x_0) + \epsilon \]
Two possibilities: Player 2:

Player 1: \(3 + (y_0 - 2x_0)\)
Player 2: \(3 - (y_0 - 2x_0) + \epsilon\)

Player 2 has a strategy to get payoff \(3 - |y_0 - 2x_0| + \epsilon\) (with \(\epsilon > 0\)) and give payoff \(3 + |y_0 - 2x_0|\) to Player 1.

There is a NE if and only if the two-counter machine halts.
What do we want to do?

- We want to use the idea of the approximation algorithm to compute \( \epsilon \)-NE (or \( \epsilon \)-subgame perfect equilibria) in weighted timed games...
What do we want to do?

- We want to use the idea of the approximation algorithm to compute $\epsilon$-NE (or $\epsilon$-subgame perfect equilibria) in weighted timed games...
- ... with the help of [BBD10,BBDG12]

Conclusion 😊

Summary of the talk

- Quick overview of results concerning the optimal reachability problem in weighted timed games
- New insight into the value problem for this model:
  - Undecidability of this problem
  - Approximability of the optimal cost (under some conditions)

Future work

- Improve the approximation scheme \(2\text{EXP}(|G|) \cdot \left(\frac{1}{\epsilon}\right)^{|X|}\)
- Extend to the whole class of weighted timed games? understand why it is not possible
- Assume stochastic uncertainty
- Is the value of any game a rational number?
- Understand the multiplayer setting