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:

$P_1$ (fast):

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

<table>
<thead>
<tr>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
</tr>
<tr>
<td>in use</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

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

<table>
<thead>
<tr>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
</tr>
<tr>
<td>in use</td>
</tr>
</tbody>
</table>

---

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>time</th>
<th>2 picoseconds</th>
<th>3 picoseconds</th>
</tr>
</thead>
</table>

| energy | idle 10 Watt | in use 90 Watts |

$P_2$ (slow):

<table>
<thead>
<tr>
<th>time</th>
<th>5 picoseconds</th>
<th>7 picoseconds</th>
</tr>
</thead>
</table>

| energy | idle 20 Watts | in use 30 Watts |

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

$P_1$ (fast):

<table>
<thead>
<tr>
<th>time</th>
<th>2 picoseconds</th>
<th>3 picoseconds</th>
</tr>
</thead>
</table>

| energy | idle 10 Watt | in use 90 Watts |

$P_2$ (slow):

<table>
<thead>
<tr>
<th>time</th>
<th>5 picoseconds</th>
<th>7 picoseconds</th>
</tr>
</thead>
</table>

| 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:

\[ \begin{align*}
P_1 \text{ (fast):} \\
\times & \quad 3 \text{ picoseconds} \\
+ & \quad 2 \text{ picoseconds} \\
\text{time} & \phantom{4} \\
\text{energy} & \\
\text{idle} & \phantom{4} 10 \text{ Watt} \\
in \text{ use} & \phantom{4} 90 \text{ Watts} \\
\end{align*} \]

\[ \begin{align*}
P_2 \text{ (slow):} \\
\times & \quad 7 \text{ picoseconds} \\
+ & \quad 5 \text{ picoseconds} \\
\text{time} & \phantom{4} \\
\text{energy} & \\
\text{idle} & \phantom{4} 20 \text{ Watts} \\
in \text{ use} & \phantom{4} 30 \text{ Watts} \\
\end{align*} \]

\[ \begin{align*}
\text{0} & \quad \text{5} & \quad \text{10} & \quad \text{15} & \quad \text{20} & \quad \text{25} \\
\text{T}_1 & \quad \text{T}_2 & \quad \text{T}_3 & \quad \text{T}_4 & \quad \text{T}_5 & \quad \text{T}_6 \\
\text{T}_1 & \quad \text{T}_2 & \quad \text{T}_3 & \quad \text{T}_4 & \quad \text{T}_5 & \quad \text{T}_6 \\
\end{align*} \]

An example: The task graph scheduling problem

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

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

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

\[
\begin{array}{|c|c|}
\hline
\text{energy} & \hline
\text{idle} & 10 \text{ Watt} \\
\text{in use} & 90 \text{ Watts} \\
\hline
\end{array}
\]

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

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

\[
\begin{array}{|c|c|}
\hline
\text{energy} & \hline
\text{idle} & 20 \text{ Watts} \\
\text{in use} & 30 \text{ Watts} \\
\hline
\end{array}
\]
The model of timed automata

done, $22 \leq y \leq 25$

problem, $x := 0$

repair, $x \leq 15$

$y := 0$

repair $2 \leq y \land x \leq 56$

$y := 0$

delayed, $y := 0$

fail-safe

$15 \leq x \leq 16$

$22 \leq y \leq 25$

fail-safe

repair

$22 \leq y \leq 25$

safe

problem, $x := 0$

repair, $x \leq 15$

$y := 0$

repair $2 \leq y \land x \leq 56$

$y := 0$

delayed, $y := 0$

fail-safe
The model of timed automata

- **safe** $\xrightarrow{23} \text{safe}$
- **problem**, $x := 0$ $\xrightarrow{} \text{alarm}$
- **alarm**, $y := 0$ $\xrightarrow{} \text{repair}$
- **repair**, $x \leq 15$ $\xrightarrow{} \text{done}$
- **done**, $22 \leq y \leq 25$ $\xrightarrow{} \text{fail-safe}$
- **fail-safe** $\xrightarrow{2.3} \text{fail-safe}$
- **repair**, $2 \leq y \land x \leq 56$ $\xrightarrow{} \text{repairing}$
- **repairing**, $y := 0$ $\xrightarrow{} \text{done}$
- **done**, $40$ $\xrightarrow{} \text{safe}$

\[ x = 0 \]
\[ y = 0 \]
\[ x = 23 \]
\[ y = 23 \]
\[ x = 0 \]
\[ y = 23 \]
\[ x = 0 \]
\[ y = 23 \]
Modelling the task graph scheduling problem
Modelling the task graph scheduling problem

- Processors

\[ P_1: \quad \text{(x} \leq 2) \quad \begin{cases} \text{idle} & \text{add}_1, \text{done}_1 \quad x = 2 \\ x := 0 & \text{mult}_1 \end{cases} \quad \begin{cases} \text{idle} & \text{add}_1, \text{done}_1 \quad x = 3 \\ x := 0 & \text{mult}_1 \end{cases} \quad (x \leq 3) \]

\[ P_2: \quad \text{(y} \leq 5) \quad \begin{cases} \text{idle} & \text{add}_2, \text{done}_2 \quad y = 5 \\ x := 0 & \text{mult}_2 \end{cases} \quad \begin{cases} \text{idle} & \text{add}_2, \text{done}_2 \quad y = 7 \\ x := 0 & \text{mult}_2 \end{cases} \quad (y \leq 7) \]
Modelling the task graph scheduling problem

- **Processors**
  - \(P_1: \) \(x = 2\) 
    - idle
    - \(x := 0\) done
    - \(x := 0\) add
    - \(x := 0\) mult
  - \(x = 3\) 
    - idle
    - \(x := 0\) done
    - \(x := 0\) mult
  - \((x \leq 2)\)
  - \((x \leq 3)\)
  - \((y \leq 5)\)
  - \((y \leq 7)\)

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

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
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 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 \[\text{AD94}\]
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

\[ x = 1 \Rightarrow x := 0 \quad y := 0 \]
\[ x \leq 2, \quad x := 0 \Rightarrow y \geq 2 \quad y := 0 \]
\[ x = 0 \land y \geq 2 \]

Theorem \[\text{[AD94]}\]
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
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
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,
  - ...

  \(\rightarrow\) 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,
  - ...

→ timed automata are not powerful enough!

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

### The thermostat example

```
\[ T \leq 19 \quad \Rightarrow \quad \dot{T} = -0.5 T \quad (T \geq 18) \]
```

```
\[ T \geq 21 \quad \Rightarrow \quad \dot{T} = 2.25 - 0.5 T \quad (T \leq 22) \]
```
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

- **Off**
  - \( \dot{T} = -0.5T \) for \( T \geq 18 \)
  - \( T \leq 19 \)
  - \( T \geq 21 \)

- **On**
  - \( \dot{T} = 2.25 - 0.5T \) for \( T \leq 22 \)
  - \( T \leq 21 \)

![Thermostat Diagram](image-url)
Ok…
Ok...

Easy...
Ok...

Easy...
Ok...
Ok... but?

Easy...

constraint

constraint

Easy...

constraint

Easy...
Ok... but?
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

---

[HKPV95] Henzinger, Kopke, Puri, Varaiya. What’s decidable wbout hybrid automata? *(SToC’95)*.

[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)*.
Modelling the task graph scheduling problem

**Processors**

- **P1**:
  - Initial state: \( x = 2 \)
  - Transition to \( x = 3 \)
  - Transition to \( \times \)

- **P2**:
  - Initial state: \( y = 5 \)
  - Transition to \( y = 7 \)
  - Transition to \( \times \)

**Tasks**

- **T4**:
  - \( t_1 \land t_2 \)
  - Transition to \( t_4 := 1 \)

- **T5**:
  - \( t_3 \)
  - Transition to \( t_5 := 1 \)
Modelling the task graph scheduling problem

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

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

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

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

\[ \ell_0 \xrightarrow{+5} \ell_1 \xrightarrow{(y=0)} \ell_2 \xrightarrow{u} \ell_3 \xrightarrow{+1} \]
Weighted/priced timed automata \[\text{[ALP01,BFH+01]}\]

\[
\begin{align*}
\ell_0 & \xrightarrow{\ell_0 + 5} \ell_1 \\
\ell_1 & \xrightarrow{u} \ell_0 \quad (y=0) \\
\ell_1 & \xrightarrow{u} \ell_2 \\
\ell_2 & \xrightarrow{x=2,c} +1 \\
\ell_3 & \xrightarrow{u} \ell_1 \\
\ell_3 & \xrightarrow{x=2,c} +7 \\
\ell_3 & \xrightarrow{c} \ell_0 \\
\ell_0 & \xrightarrow{1.3} \ell_0 \\
\ell_0 & \xrightarrow{c} \ell_1 \\
\ell_1 & \xrightarrow{u} \ell_3 \\
\ell_3 & \xrightarrow{0.7} \ell_3 \\
\ell_3 & \xrightarrow{c} \ell_0 \\
\end{align*}
\]

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

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

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

\[
\begin{align*}
\ell_0 \xrightarrow{5} & \quad x \leq 2, c, y := 0 \\
\ell_0 \quad & \quad (y = 0) \\
\ell_1 \xrightarrow{+10} & \quad u \\
\ell_2 \xrightarrow{x = 2, c} & \quad (y = 0) \\
\ell_3 \xrightarrow{u} & \quad x = 2, c \\
\ell_3 \xrightarrow{c} & \quad +1 \\
\end{align*}
\]

\[
\begin{array}{c|ccccc}
\ell & 1.3 & c & u & 0.7 & c \\
\hline
x & 0 & 1.3 & 1.3 & 1.3 & 2 \\
y & 0 & 1.3 & 0 & 0 & 0.7 \\
\end{array}
\]

cost : 

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

![Diagram of a weighted/priced timed automaton]

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

\[
\begin{align*}
x & \quad 0 & 1.3 & 1.3 & 1.3 & 2 & 2 \\
y & \quad 0 & 1.3 & 0 & 0 & 0.7 & 0.7 \\
\end{align*}
\]

cost : 6.5


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

\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 \quad 1.3 \quad 1.3 \quad 1.3 \quad 2 \\
y & 0 \quad 1.3 \quad 0 \quad 0 \quad 0.7
\end{align*}

cost : \quad 6.5 \quad + \quad 0

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

\[ \ell_0 \xrightarrow{1.3} \ell_0 \xrightarrow{c} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \ell_3 \xrightarrow{c} \smiley \]

<table>
<thead>
<tr>
<th>State</th>
<th>$x$</th>
<th>$y$</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_0$</td>
<td>0</td>
<td>0</td>
<td>6.5</td>
</tr>
<tr>
<td>$\ell_1$</td>
<td>1.3</td>
<td>1.3</td>
<td>0</td>
</tr>
<tr>
<td>$\ell_3$</td>
<td>1.3</td>
<td>1.3</td>
<td>0</td>
</tr>
</tbody>
</table>

Cost: $6.5 + 0 + 0 = 6.5$
Weighted/priced timed automata [ALP01,BFH+01]

\[
\begin{align*}
\ell_0 & \xrightarrow{1.3} \ell_0 & x \leq 2, c, y := 0 \\
\ell_1 & \xrightarrow{c} \ell_1 & (y = 0) \\
\ell_2 & \xrightarrow{u} \ell_3 \\
\ell_3 & \xrightarrow{c} \ell_3 & x = 2, c \\
\end{align*}
\]

\[\ell_0 \xrightarrow{+5} \ell_0 \Rightarrow x \leq 2, c, y := 0 \Rightarrow \ell_1 \xrightarrow{1.3} \ell_1 \xrightarrow{c} \ell_1 \xrightarrow{u} \ell_3 \xrightarrow{0.7} \ell_3 \xrightarrow{c} \text{smiley face} \]

Cost:

\[
\begin{align*}
x & = 0 & 1.3 & 1.3 & 1.3 & 2 \\
y & = 0 & 1.3 & 0 & 0 & 0.7 \\
\text{cost} & : & 6.5 & + & 0 & + & 0 & + & 0.7 \\
\end{align*}
\]

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

\[\ell_0 \xrightarrow{+5} \ell_1 \xrightarrow{u} \ell_2 \xrightarrow{+10} \ell_3 \xrightarrow{c} \ell_3 \xrightarrow{c} \ell_3 \xrightarrow{x=2,c} +1 \xrightarrow{u} \ell_2 \xrightarrow{u} \ell_1 \xrightarrow{x=2,c,y:=0} \ell_0 \]

\[
\begin{array}{cccccccc}
\ell_0 & \ell_0 & \ell_1 & \ell_3 & \ell_3 & \ell_3 & \ell_0 \\
\ell_1 & \ell_0 & \ell_1 & \ell_3 & \ell_3 & \ell_3 & \ell_0 \\
x & 0 & 1.3 & 1.3 & 1.3 & 2 & 0 \\
y & 0 & 1.3 & 0 & 0 & 0.7 & 7 \\
\end{array}
\]

cost : 6.5 + 0 + 0 + 0.7 + 7

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

\[\ell_0 \xrightarrow{+5} \ell_1 \] \(x \leq 2, c, y:=0\)

\[\ell_1 \xrightarrow{u} \ell_2 \ \ x=2, c \]
\[\ell_1 \xrightarrow{u} \ell_3 \ \ x=2, c \]

\[\ell_2 \xrightarrow{+10} \]

\[\ell_3 \xrightarrow{+1} \]

Cost:
\[
\begin{align*}
\ell_0 & \quad 1.3 & \ell_0 & \quad c & \ell_1 & \quad u & \ell_3 & \quad 0.7 & \ell_3 & \quad c & \ell_4 \\
0 & \quad 1.3 & 1.3 & \quad 1.3 & 1.3 & 2 & \quad & & & & \quad 0 \\
0 & \quad 1.3 & 0 & \quad 0 & 0 & 0.7 & \quad 7 & \quad & & & \quad 14.2 \\
\end{align*}
\]

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

Question: what is the optimal cost for reaching 😊 ?
Weighted/priced timed automata [ALP01,BFH+01]

\[
\ell_0 \quad \xrightarrow{x \leq 2, c, y := 0} \quad \ell_1 \quad \xrightarrow{u} \quad \ell_2 \quad \xrightarrow{x = 2, c} \quad \ell_3 \quad \xrightarrow{x = 2, c} \quad \text{smiley face}
\]

**Question**: what is the optimal cost for reaching \( \text{smiley face} \)?

\[
5t + 10(2 - t) + 1
\]
Weighted/priced timed automata [ALP01,BFH+01]

Question: what is the optimal cost for reaching $\square$?

$$5t + 10(2 - t) + 1, \quad 5t + (2 - t) + 7$$
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 [ALP01,BFH+01]

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

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


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

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

**Question:** what is the optimal cost for reaching \( \text{happy face} \)?

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

\( \sim \) strategy: leave immediately \( \ell_0 \), go to \( \ell_3 \), and wait there 2 t.u.

\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).
Optimal-cost reachability

Theorem \([\text{ALP01,BFH+01,BBBR07}]\)

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 \((HSCC'01)\).

[BFH+01] Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn, Vaandrager. Minimum-cost reachability in priced 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
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[ t_1 \xrightarrow{x \leq 2} t_2 \xrightarrow{} t_3 \xrightarrow{} t_4 \xrightarrow{} t_5 \xrightarrow{} \ldots \]
\[ t_1 + t_2 \leq 2 \]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[ t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow \cdots \]

\[
\begin{align*}
t_1 + t_2 & \leq 2 \\
t_2 + t_3 + t_4 & \geq 5
\end{align*}
\]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[
\begin{align*}
T_1 & \quad T_2 & \quad T_3 & \quad T_4 & \quad T_5 \\
\begin{array}{c}
t_1 \\
y := 0
\end{array} & \quad \begin{array}{c}
t_2 \\
x \leq 2
\end{array} & \quad \begin{array}{c}
t_3 \\
y \geq 5
\end{array} & \quad \begin{array}{c}
t_4 \\
t_5
\end{array} & \quad \cdots \\
\begin{array}{l}
t_1 + t_2 \leq 2 \\
t_2 + t_3 + t_4 \geq 5
\end{array} & \quad \begin{array}{c}
T_2 \leq 2 \\
T_4 - T_1 \geq 5
\end{array}
\end{align*}
\]
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[ \begin{align*}
T_1 & & T_2 & & T_3 & & T_4 & & T_5 \\
t_1 & & t_2 & & t_3 & & t_4 & & t_5 & & \ldots & \begin{cases}
t_1 + t_2 \leq 2 & T_2 \leq 2 \\
t_2 + t_3 + t_4 \geq 5 & T_4 - T_1 \geq 5
\end{cases}
\end{align*} \]

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 \( \inf_{\overline{Z}} f \) is obtained on the border of \( \overline{Z} \) with integer coordinates.
From timed to discrete behaviours

Optimal reachability as a linear programming problem

\[
\begin{align*}
T_1 & \quad T_2 & \quad T_3 & \quad T_4 & \quad T_5 \\
t_1 & \quad t_2 & \quad t_3 & \quad t_4 & \quad t_5 \\
y := 0 & \quad x \leq 2 & \quad y \geq 5 & \quad \quad & \quad \quad \\
\end{align*}
\]

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

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.

\( \forall \) 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 \( \pi \)]
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 $\mathcal{A}_{cp}$, for any $\varepsilon > 0$, 

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

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

\[
\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon \Rightarrow |\text{cost}(\Pi) - \text{cost}(\pi_\varepsilon)| < \eta
\]
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.

$$\|\Pi - \pi_\varepsilon\|_\infty < \varepsilon \Rightarrow |\text{cost}(\Pi) - \text{cost}(\pi_\varepsilon)| < \eta$$
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]
- ...

[BBL04] Bouyer, Brinksma, Larsen. Staying Alive As Cheaply As Possible (HSCC’04).
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**
  
  $P_1$:
  
  $(x \leq 2)$
  
  \[ x:=0 \]
  
  \[ x=2 \]
  
  \[ \text{idle} \]
  
  \[ \text{add}_1 \]
  
  \[ \text{done}_1 \]
  
  \[ x:=0 \]
  
  \[ x=3 \]
  
  \[ \text{idle} \]
  
  \[ \text{mult}_1 \]
  
  \[ \text{done}_1 \]
  
  $(x \leq 3)$
  
  $P_2$:
  
  $(y \leq 5)$
  
  \[ x:=0 \]
  
  \[ y=5 \]
  
  \[ \text{idle} \]
  
  \[ \text{add}_2 \]
  
  \[ \text{done}_2 \]
  
  \[ x:=0 \]
  
  \[ y=7 \]
  
  \[ \text{idle} \]
  
  \[ \text{mult}_2 \]
  
  \[ \text{done}_2 \]
  
  $(y \leq 7)$

- **Tasks**
  
  $T_4$:
  
  \[ t_1 \land t_2 \]
  
  \[ \text{add}_i \]
  
  \[ \text{done}_i \]
  
  $T_5$:
  
  \[ t_3 \]
  
  \[ \text{add}_i \]
  
  \[ \text{done}_i \]

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

- **Processors**
  - $P_1$: 
    - $x = 2$ 
      - idle 
      - done$_1$ 
      - mult$_1$ 
      - $x = 3$ 
      - done$_1$ 
      - mult$_1$ 
    - $(x \leq 2)$ 
    - $x := 0$ 
  - $P_2$: 
    - $y = 5$ 
      - idle 
      - done$_2$ 
      - mult$_2$ 
      - $y = 7$ 
      - done$_2$ 
      - mult$_2$ 
    - $(y \leq 5)$ 
    - $x := 0$ 

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

- **Modelling energy**
  - $P_1$: 
    - $x = 2$ 
      - +90 
      - idle 
      - done$_1$ 
      - mult$_1$ 
      - $x = 3$ 
      - done$_1$ 
      - mult$_1$ 
      - +90 
    - $(x \leq 2)$ 
    - $x := 0$ 
    - +10 
    - $(x \leq 3)$ 
  - $P_2$: 
    - $y = 5$ 
      - +30 
      - idle 
      - done$_2$ 
      - mult$_2$ 
      - $y = 7$ 
      - done$_2$ 
      - mult$_2$ 
      - +30 
    - $(y \leq 5)$ 
    - $x := 0$ 
    - +20 
    - $(y \leq 7)$ 

- **Modelling uncertainty**
  - $P_1$: 
    - $x \geq 1$ 
      - done$_1$ 
      - mult$_1$ 
      - $x \geq 1$ 
    - $(x \leq 2)$ 
    - $x := 0$ 
  - $P_2$: 
    - $y \geq 3$ 
      - done$_2$ 
      - mult$_2$ 
      - $y \geq 2$ 
    - $(y \leq 5)$ 
    - $x := 0$ 
  - $(x \leq 2)$ 
    - $x := 0$ 
    - $(x \leq 3)$
Modelling the task graph scheduling problem

- **Processors**

  - P₁:
    - +
    - \( x = 2 \)
    - \( x = 3 \)
    - done₁
    - \( \times \)
    - \( (x \leq 2) \)
    - \( x := 0 \)
    - \( x := 0 \)
    - \( \times \)
    - \( (x \leq 3) \)

  - \( \times \)

  - \( \times \)

- **Tasks**

  - \( T₄: \) \( t₁ \land t₂ \)
    - \( t₄ := 1 \)
    - \( \times \)
    - add₁
    - done₁
  
  - \( T₅: \) \( t₃ \)
    - \( t₅ := 1 \)
    - \( \times \)
    - add₂
    - done₂

- **Modelling energy**

  - P₁:
    - +
    - \( +90 \)
    - \( +10 \)
    - \( +90 \)
    - \( \times \)
    - \( (x \leq 2) \)
    - \( x := 0 \)
    - \( x := 0 \)

  - \( \times \)

  - \( \times \)

  - \( \times \)

  - \( \times \)

  - \( \times \)

  - \( \times \)

  - \( \times \)

- **Modelling uncertainty**

  - P₁:
    - +
    - \( x \geq 1 \)
    - \( x \geq 1 \)
    - \( \times \)
    - \( (x \leq 2) \)
    - \( x := 0 \)
    - \( x := 0 \)

  - \( \times \)

  - \( \times \)

  - \( \times \)

  - \( \times \)

  - \( \times \)

  - \( \times \)

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 😊

Diagram:
- Node $\ell_0$ with transition $x \leq 2$ leading to $\ell_1$.
- Node $\ell_1$ with transition $x \leq 1, c_1$ leading to $\ell_2$.
- Node $\ell_2$ with transition $x < 1, u_1$ leading to $\ell_3$.
- Node $\ell_3$ with transition $x \leq 1, c_3$.
- Node $\ell_1$ with transition $x \geq 1, u_3$ leading to an outcome.
- Node $\ell_2$ with transition $x \geq 2, c_4$ leading to an outcome.
An example of a timed game

Rule of the game

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

Rule of the game

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

  \[ f : \text{history} \mapsto (\text{delay, cont. transition}) \]
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 \((0.5, c_1)\)
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 \((\ell_0, 0), (0.5, c_1)\):
  \[ \leadsto \text{can be preempted by } u_2 \]
An example of a timed game

Rule of the game

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

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

A (memoryless) winning strategy

- from \((\ell_0, 0)\), play \((0.5, c_1)\)  
  \(\sim\) 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 \((0.5, c_1)\)
  \[ \sim \] can be preempted by \(u_2\)
- from \((\ell_2, *)\), play \((1 - *, c_2)\)
- 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:

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

A (memoryless) winning strategy

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

**An example of a timed game**

\[ \ell_0 \quad \ell_1 \quad \ell_2 \quad \ell_3 \]

- **\( x \leq 2 \)**
- **\( x \geq 1, u_3 \)**
- **\( x \leq 1, c_1 \)**
- **\( x \geq 2, c_4 \)**
- **\( x < 1, u_1 \)**
- **\( x < 1, u_2, x := 0 \)**

\[ \ell_0 \quad \ell_1 \quad \ell_2 \quad \ell_3 \]
An example of a timed game

Rule of the game

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

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

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.


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.

\[ \leadsto \text{classical regions are sufficient for solving such problems} \]
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.

\[ \sim \text{ classical regions are sufficient for solving such problems} \]

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

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

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

\[
\ell_0 \xrightarrow{x \leq 2, c, y := 0} \ell_1 \xrightarrow{} (y = 0) \xrightarrow{} \ell_2 \xrightarrow{x = 2, c} \ell_3 \xrightarrow{x = 2, c} \text{happy face}
\]

Question: What is the optimal cost we can ensure while reaching \( (y = 0) \)?

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

strategy: wait in \( \ell_0 \), and when \( t = \frac{24}{53} \), go to \( \ell_1 \).
A simple weighted timed game

\[
\begin{align*}
\ell_0 & \xrightarrow{+5} \ell_1 \\
& \quad \text{\(x \leq 2, c, y := 0\)} \\
\ell_1 & \xrightarrow{+10} \ell_2 \\
& \quad \text{(\(y = 0\))} \\
& \quad \text{\(x = 2, c\)} \\
\ell_2 & \xrightarrow{+1} \text{\(\star\)} \\
& \quad \text{\(x = 2, c\)} \\
\ell_3 & \xrightarrow{+1} \text{\(\star\)} \\
& \quad \text{\(x = 2, c\)}
\end{align*}
\]
A simple weighted timed game

\[ \ell_0 \xrightarrow{x \leq 2, c, y := 0} \ell_1 \]

\[ \ell_1 \xrightarrow{u} \ell_2 \]

\[ \ell_2 \xrightarrow{x = 2, c} \ell_3 \]

\[ \ell_3 \xrightarrow{u} \ell_1 \]

\[ \ell_2 \xrightarrow{u} \text{smiley} \]

Question: what is the optimal cost we can ensure while reaching \( \text{smiley} \)?
Introduction
Overview of “old” results
Some recent developments
Conclusion
Weighted timed automata
Timed games
Weighted timed games

A simple weighted timed game

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

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

\[
\begin{align*}
\max \left( 5t + 10(2 - t) + 1, \ 5t + (2 - t) + 7 \right)
\end{align*}
\]
A simple weighted timed game

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

\[ (y = 0) \]

\[ x = 2, c + 1 \]

\[ x = 2, c + 1 \]

\[ +10 \]

\[ +1 \]

\[ +1 \]

\[ +1 \]

\[ +1 \]

Question: what is the optimal cost we can ensure while reaching \( \smiley \)?

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

\[ \ell_0 \quad \rightarrow \quad x \leq 2, c, y := 0 \]
\[ \ell_1 \quad (y = 0) \quad \rightarrow \quad +1 \]
\[ \ell_2 \quad \rightarrow \quad x = 2, c \quad +10 \]
\[ \ell_3 \quad +1 \quad \rightarrow \quad x = 2, c \]

**Question:** what is the optimal cost we can ensure while reaching \( \bigcirc \)?

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

\( \sim \) **strategy:** wait in \( \ell_0 \), and when \( t = \frac{4}{3} \), go to \( \ell_1 \)
This topic has been fairly hot these last fifteen years...

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

Key: resetting the clock somehow resets the history...
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 $y = 2x$. 

$x = x_0$  
$y = y_0$  

$z = 0$  
$z = 0$  

Add$^+$($x$)  
Add$^+$($x$)  
Add$^-$($y$)  

Add$^-$($x$)  
Add$^-$($x$)  
Add$^+$($y$)  

+2  
+1
Computing the optimal cost: why is that hard?

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

\[ z = 0 \]

\[ x = x_0 \]
\[ y = y_0 \]

\[ z = 0 \]
\[ x = x_0 \]
\[ y = y_0 \]

\[ z = 0 \]

\[ \text{Add}^+(x) \]
\[ \text{Add}^+(x) \]
\[ \text{Add}^-(y) \]
\[ +2 \]

\[ \text{Add}^-(x) \]
\[ \text{Add}^-(x) \]
\[ \text{Add}^+(y) \]
\[ +1 \]

\[ \text{In } \]
\[ \text{cost } = 2x_0 + (1 - y_0) + 2 \]
Computing the optimal cost: why is that hard?

Given two clocks \( x \) and \( y \), we can check whether \( y = 2x \).

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

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

- In the green path, cost $= 2x_0 + (1 - y_0) + 2$
- In the pink path, 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 $\text{😊}$, cost = $2x_0 + (1 - y_0) + 2$

In $\text{(cmp)}$, 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** (happy face), cost = $2x_0 + (1 - y_0) + 2$
- **In** (sad face), 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 $\bigsmiley$, cost $= 2x_0 + (1 - y_0) + 2$
* In $\bigcry$, 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 $= 3$

$\Rightarrow$ 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

  $\leadsto$ 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:

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

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.

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

\[
\begin{align*}
  x=1, x:=0 & \quad \lor \quad y=1, y:=0 \\
  u:=0 & \quad \longrightarrow \quad z:=0 \\
  u=1, u:=0 & \quad (u=0)
\end{align*}
\]
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.
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 $G$ 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 $\mathcal{G}$ 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 $\bowtie c$, whether $\text{optcost}_{\mathcal{G}} \bowtie c$?
- The existence problem asks, given $\mathcal{G}$ and a threshold $\bowtie c$, whether there exists a winning strategy in $\mathcal{G}$ such that $\text{cost}(\sigma) \bowtie c$?
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 $\triangleright c$, whether $\text{optcost}_G \triangleright c$?
- The existence problem asks, given $G$ and a threshold $\triangleleft c$, whether there exists a winning strategy in $G$ such that $\text{cost}(\sigma) \triangleleft c$?

Note: These problems are distinct...
The value of the game is 0, but no strategy has cost 0.
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.
The value of the game is 1, but there is a strategy that secures cost $< 1$. 
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 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
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

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])

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
A snapshot on the undecidability proof
A snapshot on the undecidability proof

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

$\mathcal{M}$ does not halt iff the value of $G_\mathcal{M}$ is 3

Leave with cost $3 + 1/2^n$ ($n$: 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.

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...

... 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 $C$,

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

[BJM15]

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

Optimal cost is computable... when cost is strongly non-zeno. [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... but is approximable! when cost is almost-strongly non-zeno. [BJM15]

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

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

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

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

\[ |v_{\epsilon}^+ - v_{\epsilon}^-| < \epsilon \quad \text{and} \quad v_{\epsilon}^- \leq \text{optcost}_{\mathcal{G}} \leq v_{\epsilon}^+ \]
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 $G$ 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

$\sim$ 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
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: cost > 0 implies cost ≥ κ

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

\[ \text{cost} > 0 \implies \text{cost} \geq \kappa \]
Semi-unfolding

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

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

Exact computation
Approximation scheme
Approximation scheme
First step: Tree-like parts

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

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

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

\( \mapsto \) 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

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

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

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

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

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

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

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

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

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

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

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

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

\[ c \]

\[ \ell \]

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

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

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

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

\[ v' = [Y' \leftarrow 0](v + t') \]
First step: Tree-like parts

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

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

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

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

\[ \nu'' = [Y'' \leftarrow 0](v + t'') \]
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 $f_\epsilon$

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

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

2. 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 $f_\epsilon$)

$f_\epsilon$: 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 $f_\epsilon$)

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

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

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

2. 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 $f_\epsilon$)

3. 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

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

2. 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 $f_\epsilon$)

3. 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

4. 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 \((2\text{EXP}(|G|) \cdot \left(1/\epsilon\right)^{|X|^2})\)
- Extend to the whole class of weighted timed games? Or understand why it is not possible
- Assume stochastic uncertainty
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 \(2\exp(|G|) \cdot \left(\frac{1}{\epsilon}\right)^{|X|^2}\)
- Extend to the whole class of weighted timed games? Or 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 \(2\exp(|G|) \cdot \left(\frac{1}{\epsilon}\right)^{|X|^2}\)
- Extend to the whole class of weighted timed games? Or understand why it is not possible
- Assume stochastic uncertainty
- Is the value of any game a rational number?
- Understand the multiplayer setting