Optimal strategies in weighted timed games: undecidability and approximation

Nicolas Markey
LSV, CNRS & ENS Cachan & U. Paris-Saclay, France

(joint work with Patricia Bouyer and Samy Jaziri)

AVeRTS’15 workshop – Bangaluru, India
December 19, 2015
Model checking and synthesis

system

property

[http://www.embedded.com]

always $3 \leq h \leq 12$

model-checking
algorithm

oui/non
Model checking and synthesis

system

[http://www.embedded.com]

property

always $3 \leq h \leq 12$
Reasoning about real-time systems

A timed automaton is made of a transition system, a set of clocks, timing constraints on states and transitions.

Example (A computer mouse)

- idle state:
  - left button? → left click!
  - left button? → left double click!
  - right button? → right click!
  - right button? → right double click!

- left state:
  - left button? → left click!
  - left double click!

- right state:
  - right button? → right click!
  - right double click!
Reasoning about real-time systems

**Definition ([AD90])**
A timed automaton is made of
- a transition system,
- a set of clocks,
- timing constraints on states and transitions.

**Example (A computer mouse)**

- **idle**
  - left_button? → left_click!
  - left_button? → left_double_click!

- **left**
  - left_button? → left_click!
  - left_double_click!

- **right**
  - right_button? → right_click!
  - right_button? → right_double_click!

Reasoning about real-time systems

**Definition ([AD90])**

A *timed automaton* is made of

- a transition system,
- a set of clocks,

**Example (A computer mouse)**

- **left**
  - `left_button?`
  - `left_click!`
  - `left_double_click!`
- **idle**
  - `right_button?`
  - `right_click!`
- **right**
  - `right_button?`
  - `right_double_click!`

Reasoning about real-time systems

Definition ([AD90])
A timed automaton is made of
- a transition system,
- a set of clocks,
- timing constraints on states and transitions.

Example (A computer mouse)

Continuous-time semantics

Example

Theorem (~\text{AD90, ACD93, ...})

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

Example

\[
\begin{align*}
  x &= 1, \\
  y &= 0 \\
  x &\leq 2, \ y := 0 \\
  x &= 0 \land \\
  y &\geq 2, \ y := 0
\end{align*}
\]

Theorem \(\text{(AD90, ACD93, ...)}\):
Reachability in timed automata is decidable (as well as many other important properties).
Theorem (AD90, ACD93, ...):
Reachability in timed automata is decidable (as well as many other important properties).

Continuous-time semantics

Example:

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

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

\[
\begin{align*}
x & = 0 \land \\
y & \geq 2
\end{align*}
\]
Continuous-time semantics

Example

Theorem (\cite{AD90, ACD93, ...})
Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

Example

\begin{align*}
\text{x} &= 1 \\
\text{y} &= 0 \\
\text{x} &\leq 2, \text{x} := 0 \\
\text{y} &\geq 2, \text{y} := 0 \\
\text{x} &= 0 \land \\
\text{y} &\geq 2
\end{align*}

Theorem (\cite{AD90,ACD93, ...}) Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

Example

Theorem (AD90, ACD93, ...): Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

Example

Theorem ([AD90,ACD93, ...])

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

Example

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

Theorem (AD90, ACD93, ...)

Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

Example

Theorem ([AD90, ACD93, ...])
Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

Example

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

Theorem (AD90, ACD93, ...)
Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

Example

Theorem (AD90,ACD93, ...)
Reachability in timed automata is decidable (as well as many other important properties).
Continuous-time semantics

Example

\[
\begin{align*}
x &= 1, 
\quad y := 0 \\
x \leq 2, 
\quad y := 0 \\
x = 0 \land 
\quad y \geq 2
\end{align*}
\]

Theorem ([AD90, ACD93, ...])

Reachability in timed automata is decidable (as well as many other important properties).

Region automaton

Theorem: Reachability checking in timed automata is $\text{PSPACE}$-complete.
Theorem
Reachability checking in timed automata is PSPACE-complete.
Region automaton

Theorem
Reachability checking in timed automata is PSPACE-complete.
Timed games

Definition
A timed game is made of
- a timed automaton;

Example
\[
\ell_0 \xrightarrow{(x \leq 2)} \ell_1 \xrightarrow{x \geq 1} \ell_2 \xrightarrow{x \leq 1} \ell_3 \xrightarrow{x \leq 1} \\
\ell_0 \xrightarrow{x < 1} \ell_1 \xrightarrow{x \leq 1} \ell_2 \xrightarrow{x < 1} \ell_3 \\
\ell_0 \xrightarrow{x := 0} \ell_1 \xrightarrow{x < 1} \ell_2 \xrightarrow{x \leq 1} \ell_3 \\
\ell_0 \xrightarrow{x \geq 2} \ell_2 \xrightarrow{x \geq 2} \ell_3 \\
\]
Timed games

Definition

A timed game is made of
- a timed automaton;
- a partition between controllable and uncontrollable transitions.

Example

A timed game graph with transitions:
- From $\ell_0$ with condition $x \leq 2$, if $x \geq 1$, go to $\ell_1$; if $x < 1$, go to $\ell_1$ with action $x := 0$.
- From $\ell_1$ with condition $x \leq 1$, if $x \geq 2$, go to $\ell_2$; if $x < 1$, go to $\ell_2$.
- From $\ell_2$ with condition $x \leq 1$, if $x < 1$, go to $\ell_3$.
- From $\ell_0$, if $x \geq 1$, go to an unhappy state.
- From $\ell_1$, if $x \geq 1$, go to an unhappy state.
- From $\ell_2$, if $x \geq 1$, go to an unhappy state.

A memoryless strategy in $(\ell_0, x = 0)$: wait 0, go to $\ell_1$ in $(\ell_1, x = 2)$: wait until $x = 2$, go to $\ell_1$ in $(\ell_2, x \leq 1)$: wait until $x = 1$, go to $\ell_3$.
**Timed games**

**Definition**

A **timed game** is made of
- a timed automaton;
- a partition between *controllable* and *uncontrollable* transitions.

**Example**

```
ℓ₀ (x ≤ 2) → ℓ₁ (x ≥ 1)
       ^  
       v
ℓ₁ (x < 1, x := 0) → ℓ₂ (x < 1) → ℓ₃ (x ≤ 1)

a memoryless strategy

in (ℓ₀, x = 0): wait 0.5
  goto ℓ₁

in (ℓ₁, x): wait until x = 2
  goto 😊

in (ℓ₂, x ≤ 1): wait until x = 1
  goto ℓ₃

in (ℓ₃, x ≤ 1): wait until x = 1
  goto ℓ₁
```
Timed games

Theorem ([AMPS98])

Deciding the winner in a timed game (e.g. for reachability objectives) is EXPTIME-complete.

Proof

Timed games

Theorem ([AMPS98])

Deciding the winner in a timed game (e.g. for reachability objectives) is EXPTIME-complete.

Proof

Timed games

Theorem ([AMPS98])

Deciding the winner in a timed game (e.g. for reachability objectives) is EXPTIME-complete.

Timed games

Theorem ([AMPS98])

Deciding the winner in a timed game (e.g. for reachability objectives) is EXPTIME-complete.

Proof

Timed games

Theorem ([AMPS98])

Deciding the winner in a timed game (e.g. for reachability objectives) is EXPTIME-complete.

Proof

![Diagram](image)

- regions are sufficient;
- the computation terminates.

Outline of the talk
Outline of the talk
Example: task graph scheduling

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

$P_1$ (fast):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>2 ps</td>
<td>idle: 10</td>
</tr>
<tr>
<td>$\times$</td>
<td>3 ps</td>
<td>in use: 90</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>5 ps</td>
<td>idle: 20</td>
</tr>
<tr>
<td>$\times$</td>
<td>7 ps</td>
<td>in use: 30</td>
</tr>
</tbody>
</table>
Example: task graph scheduling

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 Watts
    - In use: 90 Watts

- $P_2$ (slow):
  - Time:
    - $+$: 5 picoseconds
    - $\times$: 7 picoseconds
  - Energy:
    - Idle: 20 Watts
    - In use: 30 Watts

Time:
- $T_1 = 13$ picoseconds
- $T_2 = 1.37$ nanojoules
Example: task graph scheduling

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

$P_1$ (fast):

| operation | time  
|-----------|-------|
| +         | 2 picoseconds  
| ×         | 3 picoseconds  

<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):

| operation | time  
|-----------|-------|
| +         | 5 picoseconds  
| ×         | 7 picoseconds  

<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>

1. Example task graph scheduling

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

3. $P_1$ (fast):

| operation | time  
|-----------|-------|
| +         | 2 picoseconds  
| ×         | 3 picoseconds  

<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>

4. $P_2$ (slow):

| operation | time  
|-----------|-------|
| +         | 5 picoseconds  
| ×         | 7 picoseconds  

<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>

5. 13 picoseconds 1.37 nanojoules

6. 12 picoseconds 1.39 nanojoules
Example: task graph scheduling

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

$P_1$ (fast):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
<td>2 picoseconds</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>$\times$</td>
<td>3 picoseconds</td>
<td>$\times$</td>
</tr>
<tr>
<td>energy</td>
<td>idle</td>
<td>10 Watt</td>
<td>idle</td>
</tr>
<tr>
<td></td>
<td>in use</td>
<td>90 Watts</td>
<td>in use</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
<td>5 picoseconds</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>$\times$</td>
<td>7 picoseconds</td>
<td>$\times$</td>
</tr>
<tr>
<td>energy</td>
<td>idle</td>
<td>20 Watts</td>
<td>idle</td>
</tr>
<tr>
<td></td>
<td>in use</td>
<td>30 Watts</td>
<td>in use</td>
</tr>
</tbody>
</table>

Sch1

<table>
<thead>
<tr>
<th>Sch1</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13 picoseconds 1.37 nanojoules

Sch2

<table>
<thead>
<tr>
<th>Sch2</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12 picoseconds 1.39 nanojoules

Sch3

<table>
<thead>
<tr>
<th>Sch3</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

19 picoseconds 1.32 nanojoules
Linear hybrid automata

Definition
A linear hybrid automaton is made of
• a timed automaton;

Example

![Diagram]

- **wait**
  - \( x=1, \text{stop} \)
- **exec**
  - \( \text{start}, x:=0 \)
- **paused**
  - \( \text{pause, resume} \)
Linear hybrid automata

Definition
A linear hybrid automaton is made of
- a timed automaton;
- for each location, the rate of each clock.

Example

\[
\begin{aligned}
\text{wait: } & \dot{w} = 10 \\
\text{exec: } & \dot{w} = 90 \\
\text{paused: } & \dot{x} = 0, \dot{w} = 10 \\
\text{start, } x := 0 & \\
\text{x=1, stop } & \\
\text{pause } & \\
\text{resume } &
\end{aligned}
\]
Reachability in linear hybrid automata

Theorem ([Čer92])

Reachability in linear hybrid automata is undecidable.

Reachability in linear hybrid automata

**Theorem ([Čer92])**

*Reachability in linear hybrid automata is *undecidable*. *

**Proof**

Encode a **two-counter machine using four stopwatches**:

\[
\begin{align*}
  c_1 &= a_1 - b_1 \\
  c_2 &= a_2 - b_2 \\
  \dot{b}_1 &= 0 \\
  \dot{a}_2 &= 0 \\
  t &= 0 \\
  t &= 1, t := 0 \\
  a_2 - b_2 &= 0
\end{align*}
\]

Already undecidable for one stopwatch and no diagonal constraints.

Reachability in linear hybrid automata

Theorem ([Čer92])

Reachability in linear hybrid automata is **undecidable**.

**Proof**

Encode a **two-counter machine using four stopwatches**:

\[
\begin{align*}
    c_1 &= a_1 - b_1 \\
    c_2 &= a_2 - b_2
\end{align*}
\]

Reachability in linear hybrid automata

Theorem ([Čer92])

Reachability in linear hybrid automata is undecidable.

Proof

Encode a two-counter machine using four stopwatches:

\[
\begin{align*}
  c_1 &= a_1 - b_1 \\
  c_2 &= a_2 - b_2
\end{align*}
\]

\begin{align*}
  &c_1++ \\
  &c_2 > 0 \\
  &c_2 == 0 \\
  &t == 0 \\

  &c_2 == 0 \\
  &c_2 -- \\
  &t == 0 \\

  &t == 1, t := 0 \\
  &b_1 == 0 \\

  &t == 0 \\
  &a_2 - b_2 > 0 \\
  &a_2 == 0
\end{align*}

Already undecidable for one stopwatch and no diagonal constraints.

Theorem ([Čer92])

Reachability in linear hybrid automata is \textit{undecidable}.

Proof

Encode a \textbf{two-counter machine using four stopwatches:}

\begin{align*}
c_1 &= a_1 - b_1 \\
c_2 &= a_2 - b_2
\end{align*}

Already undecidable for \textbf{one stopwatch and no diagonal constraints}.
Priced timed automata

Definition ([KPSY99, ALP01, BFH+01])

A priced timed automaton is made of

- a timed automaton;

Example

![Diagram of priced timed automaton]

Priced timed automata

Definition ([KPSY99, ALP01, BFH+01])

A priced timed automaton is made of
- a timed automaton;
- the price of each transition and location.

Example

\[
\begin{align*}
&x := 0 & -1 \\
&-6 & x = 1 & +2 & +6 \\
&-3 &
\end{align*}
\]

Priced timed automata

Definition ([KPSY99, ALP01, BFH+01])

A priced timed automaton is made of
- a timed automaton;
- the price of each transition and location.

Example

Priced timed automata

Definition ([KPSY99, ALP01, BFH+01])

A priced timed automaton is made of

- a timed automaton;
- the price of each transition and location.

Example

\[\begin{align*}
-3 & \xrightarrow{x:=0} -6 \xrightarrow{x=1} -3 \\
+6 & \xrightarrow{-1} +2 \\
\end{align*}\]

Priced timed automata

Definition ([KPSY99, ALP01, BFH+01])

A priced timed automaton is made of
- a timed automaton;
- the price of each transition and location.

Example

![Graph example](image)

Priced timed automata

Definition ([KPSY99, ALP01, BFH+01])

A priced timed automaton is made of
- a timed automaton;
- the price of each transition and location.

Example

Priced timed automata

Definition ([KPSY99, ALP01, BFH+]01])
A priced timed automaton is made of
- a timed automaton;
- the price of each transition and location.

Example

[3]
[6]
x:=0

\[\begin{align*}
-3 & \rightarrow +6 \\
-6 & \rightarrow +2 \\
-3 & \rightarrow -3 \\
+6 & \rightarrow +6 \\
-6 & \rightarrow -1 \\
\end{align*}\]

[0] [1] [2] [3] [4]

Priced timed automata

Definition ([KPSY99, ALP01, BFH+01])

A priced timed automaton is made of

- a timed automaton;
- the price of each transition and location.

Example

\[ \begin{align*}
-3 & \quad \rightarrow \quad +6 \\
-6 & \quad \rightarrow \quad 0 & x := 0 \\
-1 & \quad \rightarrow \quad +2 & x = 1 \\
-3 & \quad \rightarrow \quad -3 & \frac{1}{6} \\
+6 & \quad \rightarrow \quad +6 & \frac{1}{2} \\
+6 & \quad \rightarrow \quad -1 \\
-6 & \quad \rightarrow \quad -6 & \frac{1}{3}
\end{align*} \]

Priced timed automata

**Definition ([KPSY99, ALP01, BFH+01])**

A *priced timed automaton* is made of
- a timed automaton;
- the *price* of each transition and location.

**Example**

![Example Diagram]

-3 \[\xrightarrow{\frac{1}{6}}\] -3 \[\xrightarrow{\frac{1}{6}}\] +6 \[\xrightarrow{\frac{1}{2}}\] +6 \[\xrightarrow{-1}\] -6 \[\xrightarrow{\frac{1}{3}}\] -6 \[\xrightarrow{2}\] +2

---

Priced timed automata

Definition ([KPSY99, ALP01, BFH+01])

A priced timed automaton is made of
- a timed automaton;
- the price of each transition and location.

Example

Example: task graph scheduling

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>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>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>

$T_1$ 13 picoseconds 1.37 nanojoules
$T_2$ 12 picoseconds 1.39 nanojoules
$T_3$ 19 picoseconds 1.32 nanojoules
Modelling the task graph scheduling problem

Processors:

- \[ c = 90 \]
- \[ x \leq 2 \]
- done_1
- \[ c = 10 \]
- \[ x = 2 \]
- add_1
- \[ x = 0 \]
- idle
- done_1
- \[ c = 90 \]
- \[ x = 3 \]
- mul_1
- \[ x = 0 \]
- \[ x \leq 3 \]
- \times
Modelling the task graph scheduling problem

Processors:

```
+   \( \dot{c} = 90 \)   \( \dot{c} = 30 \)
\( x \leq 2 \)   \( x \leq 5 \)

\( \text{add}_1 \)   \( \text{add}_2 \)   \( \text{done}_1 \)   \( \text{done}_2 \)
\( x : = 0 \)   \( x : = 0 \)   \( x : = 0 \)   \( x : = 0 \)

idle   \( \dot{c} = 10 \)   \( \dot{c} = 20 \)
\( x \leq 3 \)   \( x \leq 7 \)

\( \times \)   \( \dot{c} = 90 \)   \( \dot{c} = 30 \)
\( x \leq 3 \)   \( x \leq 7 \)

\( \text{mul}_1 \)   \( \text{mul}_2 \)
\( x : = 0 \)   \( x : = 0 \)
```
Modelling the task graph scheduling problem

Processors:

Tasks:
Outline of the talk
Cost-optimal reachability in priced timed automata

Example

\[
\begin{align*}
\dot{p} &= 5 \\
\dot{p} &= 6 \\
\dot{p} &= 3 \\
\end{align*}
\]

\[x \leq 2, y := 0, x \geq 3, p += 1, p += 9\]

Minimal cost for reaching \(y = 0\):

\[
\inf_{0 \leq t \leq 2} \min \left(5t + 6(3 - t) + 1, 5t + 3(3 - t) + 9\right) = 17
\]

The optimal schedule consists in waiting 2 time units; going through.
Cost-optimal reachability in priced timed automata

Example

\[ \dot{p} = 5 \quad \dot{y} = 0 \quad x \leq 2 \quad y := 0 \quad y = 0 \quad \dot{p} = 3 \quad \dot{p} = 6 \quad p += 1 \quad x \geq 3 \quad p += 9 \quad x \geq 3 \]

Minimal cost for reaching \( \smiley \):

\[
\inf_{0 \leq t \leq 2} \min (5t + 6(3 - t) + 1, 5t + 3(3 - t) + 9) = 17
\]
Cost-optimal reachability in priced timed automata

Example

\[ \dot{p} = 5 \]
\[ x \leq 2, \quad y := 0 \]
\[ y = 0 \]

\[ \dot{p} = 6 \]
\[ x \geq 3, \quad p += 1 \]

\[ \dot{p} = 3 \]
\[ x \geq 3, \quad p += 9 \]

Minimal cost for reaching \( \mathbb{E} \):
\[ 5t + 6(3 - t) + 1 \]
Cost-optimal reachability in priced timed automata

Example

\[ \dot{p} = 5 \quad \begin{array}{c} x \leq 2 \\ y := 0 \end{array} \]

\[ \dot{p} = 0 \]

\[ \dot{p} = 6 \quad x \geq 3 \quad p += 1 \]

\[ \dot{p} = 3 \quad x \geq 3 \quad p += 9 \]

Minimal cost for reaching \( \smile \):

\[
5t + 6(3 - t) + 1 \\
5t + 3(3 - t) + 9
\]
Cost-optimal reachability in priced timed automata

Example

\[ \dot{p} = 5 \]
\[ x \leq 2 \]
\[ y := 0 \]

\[ y = 0 \]

\[ \dot{p} = 6 \]
\[ x \geq 3 \]
\[ p + = 1 \]

\[ \dot{p} = 3 \]
\[ x \geq 3 \]
\[ p + = 9 \]

Minimal cost for reaching \(\smile\):

\[
\min \left( 5t + 6(3 - t) + 1 \right) \left( 5t + 3(3 - t) + 9 \right)
\]
Cost-optimal reachability in priced timed automata

Example

\[ \dot{p} = 5 \] \hspace{1cm} x \leq 2 \hspace{1cm} y := 0 \hspace{1cm} \dot{y} = 0 \hspace{1cm} y = 0 \hspace{1cm} \dot{p} = 3 \hspace{1cm} x \geq 3 \hspace{1cm} p += 1 \hspace{1cm} \dot{p} = 6 \hspace{1cm} x \geq 3 \hspace{1cm} p += 9 \]

Minimal cost for reaching 😊:

\[ \inf_{0 \leq t \leq 2} \min \left( \begin{array}{l}
5t + 6(3 - t) + 1 \\
5t + 3(3 - t) + 9
\end{array} \right) \]

\[ = 17 \]
Cost-optimal reachability in priced timed automata

Example

\[ \dot{p} = 5 \quad x \leq 2 \quad y := 0 \quad y = 0 \]

\[ \dot{p} = 6 \quad p += 1 \quad x \geq 3 \]

\[ \dot{p} = 3 \quad p += 9 \quad x \geq 3 \]

Minimal cost for reaching \( \ 🙂 \):

\[
\inf_{0 \leq t \leq 2} \min \left( \begin{array}{c}
5t + 6(3 - t) + 1 \\
5t + 3(3 - t) + 9
\end{array} \right) = 17
\]

The optimal schedule consists in waiting 2 time units in; going through.
Cost-optimal reachability in priced timed automata

Example

\[ \dot{p} = 5 \]

\[ x \leq 2 \]

\[ y := 0 \]

\[ y = 0 \]

\[ \dot{p} = 6 \]

\[ x \geq 3 \]

\[ p + = 1 \]

\[ \dot{p} = 3 \]

\[ x \geq 3 \]

\[ p + = 9 \]

Minimal cost for reaching 😊:

\[
\inf_{0 \leq t \leq 2} \min \left( \begin{array}{c}
5t + 6(3 - t) + 1 \\
5t + 3(3 - t) + 9
\end{array} \right) = 17
\]

The optimal schedule consists in

- waiting 2 time units in 😏;
- going through 😄.
Cost-optimal reachability in priced timed automata

Theorem ([BBBR07])

*Optimal reachability in priced timed automata is PSPACE-complete.*

Theorem ([BBBR07])

Optimal reachability in priced timed automata is PSPACE-complete.

Proof
- Regions are not precise enough;

Cost-optimal reachability in priced timed automata

**Theorem ([BBBR07])**

*Optimal reachability in priced timed automata is PSPACE-complete.*

**Proof**
- Regions are **not precise enough**;
- Use regions with **corner-points**;

Cost-optimal reachability in priced timed automata

**Theorem ([BBBR07])**

*Optimal reachability in priced timed automata is PSPACE-complete.*

**Proof**

- Regions are not precise enough;
- Use regions with corner-points:

Cost-optimal reachability in priced timed automata

**Theorem ([BBBR07])**

**Optimal reachability in priced timed automata is PSPACE-complete.**

**Proof**

- Regions are **not precise enough**;
- Use regions with **corner-points**:

Theorem ([BBBR07])

Optimal reachability in priced timed automata is PSPACE-complete.

Proof

- Regions are not precise enough;
- Use regions with corner-points:

Outline of the talk
Example: task graph scheduling

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

$P_1$ (fast):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th></th>
<th></th>
<th>time</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>2 ps</td>
<td></td>
<td>×</td>
<td>3 ps</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>energy</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></th>
<th>time</th>
<th></th>
<th></th>
<th>time</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>5 ps</td>
<td></td>
<td>×</td>
<td>7 ps</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>energy</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>

Schedulings:

<table>
<thead>
<tr>
<th></th>
<th>T₁</th>
<th>T₂</th>
<th>T₃</th>
<th>T₄</th>
<th>T₅</th>
<th>T₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sch₁</td>
<td></td>
<td>T₂</td>
<td>T₃</td>
<td>T₅</td>
<td>T₆</td>
<td></td>
</tr>
<tr>
<td>P₁</td>
<td>T₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>T₁</th>
<th>T₃</th>
<th>T₅</th>
<th>T₄</th>
<th>T₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sch₂</td>
<td>T₁</td>
<td>T₃</td>
<td>T₅</td>
<td>T₄</td>
<td>T₆</td>
</tr>
<tr>
<td>P₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P₂</td>
<td>T₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>T₁</th>
<th>T₃</th>
<th>T₄</th>
<th>T₅</th>
<th>T₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sch₃</td>
<td>T₁</td>
<td>T₃</td>
<td>T₄</td>
<td></td>
<td>T₆</td>
</tr>
<tr>
<td>P₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P₂</td>
<td>T₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13 picoseconds 1.37 nanojoules
12 picoseconds 1.39 nanojoules
19 picoseconds 1.32 nanojoules
Cost-optimal reachability in priced timed games

Using games to model uncertainty over delays

Processors with exact delays:

\[ \dot{c} = 90, \quad x \leq 2 \]

\[ \dot{c} = 10, \quad x = 0 \]

\[ \dot{c} = 90, \quad x \leq 3 \]
Cost-optimal reachability in priced timed games

Using games to model uncertainty over delays

Processors with exact delays:

Processors with approximate delays:
Cost-optimal reachability in priced timed games

Example

\[ \dot{p} = 5 \quad y := 0 \quad x \leq 2 \]

\[ \dot{p} = 3 \quad x \geq 3 \quad p += 9 \]

\[ \dot{p} = 6 \quad x \geq 3 \quad p += 1 \]

Infimum cost for reaching: \[ \inf_{0 \leq t \leq 2} \max \left( 5t + 6(3 - t) + 1, 5t + 3(3 - t) + 9 \right) = 18.66 \quad (with \quad t_{opt} = 1.3) \]
Cost-optimal reachability in priced timed games

Example

Minimal cost for reaching 😊:

\[
\inf_{0 \leq t \leq 2} \max(5t + 6(3 - t) + 1, 5t + 3(3 - t) + 9) = 18.66
\]

(with \(t_{opt} = 1.3\))
Cost-optimal reachability in priced timed games

Example

\[ \begin{align*}
\dot{p} &= 5 \\
\dot{x} &= 2 \\
y &= 0 \\
p &= +1 \\
\end{align*} \]

Minimal cost for reaching \( \smiley \):

\[ 5t + 6(3 - t) + 1 \]
Cost-optimal reachability in priced timed games

Example

\[ \dot{p} = 5 \]

\[ \dot{p} = 6 \]

\[ \dot{p} = 3 \]

\[ x \leq 2 \]

\[ y := 0 \]

\[ y = 0 \]

\[ x \geq 3 \]

\[ p += 1 \]

\[ p += 9 \]

Minimal cost for reaching 😊:

\[ 5t + 6(3 - t) + 1 \]

\[ 5t + 3(3 - t) + 9 \]
Cost-optimal reachability in priced timed games

Example

\[ \dot{p} = 5 \quad x \leq 2 \quad y := 0 \quad \dot{y} = 0 \quad y = 0 \]

\[ \dot{p} = 6 \quad x \geq 3 \quad \dot{p} += 1 \]

\[ \dot{p} = 3 \quad x \geq 3 \quad \dot{p} += 9 \]

Minimal cost for reaching 😊:

\[
\max \left( 5t + 6(3 - t) + 1, \ 5t + 3(3 - t) + 9 \right)
\]

\[
\inf_{0 \leq t \leq 2} \max (5t + 6(3 - t) + 1, 5t + 3(3 - t) + 9) = 18.66
\]

(with \( t_{opt} = \frac{1}{3} \))
Cost-optimal reachability in priced timed games

Example

\[ \dot{p} = 5 \]
\[ x \leq 2 \]
\[ y := 0 \]
\[ y = 0 \]
\[ \dot{p} = 6 \]
\[ x \geq 3 \]
\[ p += 1 \]
\[ \dot{p} = 3 \]
\[ x \geq 3 \]
\[ p += 9 \]

Minimal cost for reaching 😊:

\[
\inf_{0 \leq t \leq 2} \max \left( \begin{array}{l}
5t + 6(3 - t) + 1 \\
5t + 3(3 - t) + 9
\end{array} \right)
\]

\[ = 18.66 \] (with \( t_{opt} = \frac{1}{3} \))
Cost-optimal reachability in priced timed games

Example

\[ \dot{p} = 5 \quad x \leq 2 \quad y := 0 \quad y = 0 \quad \dot{p} = 6 \quad x \geq 3 \quad p += 1 \quad \dot{p} = 3 \quad x \geq 3 \quad p += 9 \]

Minimal cost for reaching ☺:

\[ \inf_{0 \leq t \leq 2} \max \left( \begin{array}{c} 5t + 6(3 - t) + 1 \\ 5t + 3(3 - t) + 9 \end{array} \right) = 18.66 \]
Cost-optimal reachability in priced timed games

Example

\[
\begin{align*}
\dot{p} &= 5 \\
x \leq 2 & \quad y := 0 \\
y &= 0 & \quad \dot{p} = 6 \\
x \geq 3 & \quad p += 1 \\
\dot{p} &= 3 \\
x \geq 3 & \quad p += 9
\end{align*}
\]

Minimal cost for reaching \( \smiley \):

\[
\inf_{0 \leq t \leq 2} \max \left( \begin{array}{c}
5t + 6(3 - t) + 1 \\
5t + 3(3 - t) + 9
\end{array} \right) = 18.66
\]

(with \( t_{\text{opt}} = \frac{1}{3} \))
Cost-optimal reachability in priced timed games

Theorem ([BBR05,BBM06])

*Optimal reachability in priced timed games is *undecidable.*

Wouldn't almost-optimal strategies be sufficient?

Theorem ([BBR05,BBM06])

Optimal reachability in priced timed games is undecidable.

Proof

Encode a two-counter machine as a priced timed game.

Theorem ([BBR05,BBM06])

Optimal reachability in priced timed games is undecidable.

Proof

Encode a two-counter machine as a priced timed game.

- add the value of clock $x$ to the accumulated cost

Cost-optimal reachability in priced timed games

Theorem ([BBR05,BBM06])

*Optimal reachability in priced timed games is undecidable.*

Proof

Encode a *two-counter machine* as a priced timed game.

- add the value of clock $x$ to the accumulated cost
- add $1 - x$ to the accumulated cost

Wouldn’t almost-optimal strategies be sufficient?


Theorem ([BBR05,BBM06])

Optimal reachability in priced timed games is undecidable.

Proof

Encode a two-counter machine as a priced timed game.

- add the value of clock $x$ to the accumulated cost
- add $1 - x$ to the accumulated cost
- check that $y = 2x$
Cost-optimal reachability in priced timed games

Theorem ([BBR05, BBM06])

**Optimal reachability in priced timed games is undecidable.**

**Proof**

Encode a **two-counter machine** as a **priced timed game**.

- add the value of clock $x$ to the accumulated cost
- add $1 - x$ to the accumulated cost
- check that $y = 2x$

Wouldn’t almost-optimal strategies be sufficient?


Cost-optimal reachability in priced timed games

**Theorem ([BBR05,BBM06])**

*Optimal reachability in priced timed games is undecidable.*

**Proof**

Encode a *two-counter machine* as a priced timed game.

- add the value of clock $x$ to the accumulated cost
- add $1 - x$ to the accumulated cost
- check that $y = 2x$
- divide clock $x$ by 2

Where $\dot{p} = 0$,

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

Wouldn’t almost-optimal strategies be sufficient?

---


Cost-optimal reachability in priced timed games

Theorem ([BBR05,BBM06])

Optimal reachability in priced timed games is undecidable.

Proof

Encode a two-counter machine as a priced timed game.

- add the value of clock $x$ to the accumulated cost
- add $1 - x$ to the accumulated cost
- check that $y = 2x$
- divide clock $x$ by 2

$\leadsto$ We can use the following encoding:

$$x_1 = \frac{1}{2^{c_1}} \quad \quad x_2 = \frac{1}{2^{c_2}}$$
Cost-optimal reachability in priced timed games

Theorem ([BBR05, BBM06])

Optimal reachability in priced timed games is undecidable.

Proof

Encode a two-counter machine as a priced timed game.

Wouldn’t almost-optimal strategies be sufficient?


Cost-optimal reachability in priced timed games

Theorem ([BBR05,BBM06])

*Optimal reachability in priced timed games is undecidable.*

Proof

Encode a *two-counter machine* as a *priced timed game*.

Lemma

*The halting state is reachable if, and only if,*

*there is an optimal strategy in the priced timed game.*


Cost-optimal reachability in priced timed games

Theorem ([BBR05, BBM06])

Optimal reachability in priced timed games is undecidable.

Proof

Encode a two-counter machine as a priced timed game.

Lemma

The halting state is reachable if, and only if, there is an optimal strategy in the priced timed game.

reach terminal location with total weight at most 3


Cost-optimal reachability in priced timed games

Theorem ([BBR05, BBM06])

Optimal reachability in priced timed games is undecidable.

Proof

Encode a two-counter machine as a priced timed game.

Lemma

The halting state is reachable if, and only if, there is an optimal strategy in the priced timed game.

reach terminal location with total weight at most 3

Wouldn’t almost-optimal strategies be sufficient?

The value of a game

Definition

Cost of a path: $\text{cost}(\pi) = \text{sum of costs of all transitions until target location}$

Cost of a strategy: $\text{cost}(\sigma) = \sup\{\text{cost}(\pi) | \pi \text{ outcome of } \sigma\}$

Optimal cost in a priced timed game: $\text{optcost}_G = \inf\{\text{cost}(\sigma) | \sigma \text{ winning strategy in } G\}$

The existence of a strategy with cost less than $k$ is undecidable.

What about deciding if $\text{optcost}_G \leq k$?
The value of a game

Definition

Cost of a path:

\[ \text{cost}(\pi) = \text{sum of costs of all transitions until target location} \]
The value of a game

Definition

Cost of a path:

\[ \text{cost}(\pi) = \text{sum of costs of all transitions until target location} \]

Cost of a strategy:

\[ \text{cost}(\sigma) = \sup \{ \text{cost}(\pi) \mid \pi \text{ outcome of } \sigma \} \]
The value of a game

**Definition**

**Cost of a path:**

\[
\text{cost}(\pi) = \text{sum of costs of all transitions until target location}
\]

**Cost of a strategy:**

\[
\text{cost}(\sigma) = \sup \{ \text{cost}(\pi) \mid \pi \text{ outcome of } \sigma \}
\]

**Optimal cost in a priced timed game:**

\[
\text{optcost}_G = \inf \{ \text{cost}(\sigma) \mid \sigma \text{ winning strategy in } G \}
\]

The existence of a strategy with cost less than \(k\) is undecidable. What about deciding if \(\text{optcost}_G \leq k\)?
The value of a game

**Definition**

**Cost of a path:**

\[ \text{cost}(\pi) = \text{sum of costs of all transitions until target location} \]

**Cost of a strategy:**

\[ \text{cost}(\sigma) = \sup\{\text{cost}(\pi) \mid \pi \text{ outcome of } \sigma\} \]

**Optimal cost in a priced timed game:**

\[ \text{optcost}_G = \inf\{\text{cost}(\sigma) \mid \sigma \text{ winning strategy in } G\} \]

The existence of a strategy with cost less than \( k \) is **undecidable**.

What about deciding if \( \text{optcost}_G \leq k \)?
Undecidability of the value problem

Trying to reuse the previous reduction...

![Diagram](https://via.placeholder.com/150)

- Initial cost: 3
- The value of the game is 3, but there is no optimal strategy...
Undecidability of the value problem

Trying to reuse the previous reduction...

The value of the game is 3, but there is no optimal strategy...
Undecidability of the value problem

Trying to reuse the previous reduction...

The value of the game is 3, but there is no optimal strategy...
Undecidability of the value problem

Trying to reuse the previous reduction...

\[q_0 \quad q_1 \quad q_2 \quad q_3 \quad q_4 \quad q_5 \quad q_8 \quad q_9\]

\[c_1 += 2 \quad c_2 -= \quad c_1 -= \quad c_2 += 2\]

\[c_1 > 0 \quad c_2 > 0 \quad c_2 += 2 \quad c_1 += 2 \quad c_2 -= \quad c_1 -= \]

\[c_2 == 0 \quad c_1 += 0 \quad c_2 += 2 \quad c_1 += 2 \quad c_2 == 0 \quad c_1 += 2 \]

\[\text{final cost:} \quad 3 + \left| \left| 2 \cdot \frac{1}{25} - \frac{1}{25} \right| \right| \]

The value of the game is 3, but there is no optimal strategy...
Undecidability of the value problem

Trying to reuse the previous reduction...

The value of the game is 3, but there is no optimal strategy...

final cost: \(3 + \left| 2 \cdot \frac{1}{25} - \frac{1}{25} \right|\)
Undecidability of the value problem

Trying to reuse the previous reduction...

The value of the game is 3, but there is no optimal strategy...
Undecidability of the value problem

Adapting the previous reduction...

If $M$ does not halt:
Player 1 simulates correctly until $2^n > \epsilon$.

Cost $\sigma \leq 3 + \epsilon$.

If $M$ halts:
Correct simulation for finite duration.

Cost $\sigma \geq 3 + \alpha_M$ for all $\sigma$. 

Exit nodes:
$\sigma_{halt}$.
Undecidability of the value problem

Adapting the previous reduction...

$q_{halt}$

If $M$ does not halt:
- Player 1 simulates correctly until $2^n > 1 + \epsilon$.
- Cost($\sigma$) $\leq 3 + \epsilon$

If $M$ halts:
- Correct simulation for finite duration.
- Cost($\sigma$) $\geq 3 + \alpha_M$ for all $\sigma$
Undecidability of the value problem

Adapting the previous reduction...

exit nodes: cost $3 + \frac{1}{2^n}$
($n =$ length of path)
Undecidability of the value problem

Adapting the previous reduction...

If $M$ does not halt:
- Player 1 simulates correctly until $2^n > 1 + \epsilon$.
- $	ext{cost}(\sigma) \leq 3 + \epsilon$.

If $M$ halts:
- Correct simulation for finite duration.
- $	ext{cost}(\sigma) \geq 3 + \alpha_M$ for all $\sigma$.

exit nodes: cost $3 + \frac{1}{2^n}$

($n = \text{length of path}$)
Undecidability of the value problem

Adapting the previous reduction...

- if $M$ does not halt: Player 1 simulates correctly until $2^n > \frac{1}{\epsilon}$.

$\leadsto \text{cost}(\sigma) \leq 3 + \epsilon$

exit nodes: cost $3 + \frac{1}{2^n}$

($n = \text{length of path}$)
Undecidability of the value problem

Adapting the previous reduction...

- **if $\mathcal{M}$ does not halt:**
  Player 1 simulates correctly until $2^n > \frac{1}{\epsilon}$.

  \[ \sim \text{cost}(\sigma) \leq 3 + \epsilon \]

- **if $\mathcal{M}$ halts:**
  correct simulation for finite duration.

  \[ \sim \text{cost}(\sigma) \geq 3 + \alpha_\mathcal{M} \]
  for all $\sigma$

**exit nodes:** cost $3 + \frac{1}{2^n}$

($n = \text{length of path}$)
Undecidability of the value problem

Theorem ([BJM15])

The value problem is undecidable in priced timed games.

**Undecidability of the value problem**

**Theorem ([BJM15])**

The value problem is **undecidable** in priced timed games.

**Remark**

- **blue nodes** and intermediary instruction modules have cost zero everywhere;
- positive weights only occur in acyclic parts.

Approximation of the optimal cost

Definition

A priced timed game $G$ is almost-strongly non-Zeno if there exists $\kappa > 0$ for any run $\rho$ that starts and ends in the same region:

$$\text{cost}(\rho) \geq \kappa \quad \text{or} \quad \text{cost}(\rho) = 0$$
Definition
A priced timed game $\mathcal{G}$ is \textbf{almost-strongly non-Zeno} if there exists $\kappa > 0$ for any run $\rho$ that starts and ends in the same region:
\[
\text{cost}(\rho) \geq \kappa \quad \text{or} \quad \text{cost}(\rho) = 0
\]

Theorem ([BJM15])

The \textit{optimal cost} of almost-strongly non-Zeno priced timed automata can be \textit{approximated}.
Approximation of the optimal cost

**Definition**

A priced timed game $G$ is **almost-strongly non-Zeno** if there exists $\kappa > 0$ for any run $\rho$ that starts and ends in the same region:

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

**Theorem ([BJM15])**

The *optimal cost* of almost-strongly non-Zeno priced timed automata can be **approximated**: for every $\epsilon > 0$, we can compute

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

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

- a strategy $\sigma_\epsilon$ such that

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

Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)

```
Hypothesis: cost > 0
\downarrow
cost \geq \kappa;

compute exact optimal cost in tree-like parts [LMM02, ABM04]
compute approximate optimal cost in kernels
```
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)

```
Hypothesis: cost > 0
↓
cost ≥ κ;
bounded depth
compute exact optimal cost in tree-like parts [LMM02, ABM04]
compute approximate optimal cost in kernels
```
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)

Hypothesis: \( \text{cost} > 0 \)

Compute exact optimal cost in tree-like parts \([LMM02, ABM04]\)

Compute approximate optimal cost in kernels
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)

Hypothesis:
\[
\text{cost} > 0 \\
\downarrow \\
\text{cost} \geq \kappa
\]
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)

Hypothesis:

- cost > 0
  - cost ≥ κ
  - bounded depth

Kernel \( K \)

Only cost 0
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)
- compute **exact** optimal cost in tree-like parts [LMM02,ABM04]

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-Reachability and Control for Acyclic WTA. TCS, 2002
[ABM04] Alur, Bernadsky, Madhusudan. Optimal Reachability for Weighted Timed Games. ICALP, 2004
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)
- compute **exact** optimal cost in tree-like parts [LMM02,ABM04]

[1] La Torre, Mukhopadhyay, Murano. Optimal-Reachability and Control for Acyclic WTA. TCS, 2002
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)
- compute **exact** optimal cost in tree-like parts [LMM02,ABM04]

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-Reachability and Control for Acyclic WTA. TCS, 2002
[ABM04] Alur, Bernadsky, Madhusudan. Optimal Reachability for Weighted Timed Games. ICALP, 2004
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)
- compute **exact** optimal cost in tree-like parts [LMM02, ABM04]

[LMM02] La Torre, Mukhopadhyay, Murano. Optimal-Reachability and Control for Acyclic WTA. TCS, 2002
[ABM04] Alur, Berndsky, Madhusudan. Optimal Reachability for Weighted Timed Games. ICALP, 2004
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)
- compute **exact** optimal cost in tree-like parts [LMM02,ABM04]
- compute **approximate** optimal cost in kernels

Output cost functions $f$
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)
- compute **exact** optimal cost in tree-like parts [LMM02,ABM04]
- compute **approximate** optimal cost in kernels

Output cost functions $f$

Under- and over-approximate by piecewise constant functions $f_\epsilon^-$ and $f_\epsilon^+$
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)
- compute **exact** optimal cost in tree-like parts [LMM02,ABM04]
- compute **approximate** optimal cost in kernels

Output cost functions $f$

Under- and over-approximate by piecewise constant functions $f_\epsilon^-$ and $f_\epsilon^+$
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)
- compute **exact** optimal cost in tree-like parts [LMM02,ABM04]
- compute **approximate** optimal cost in kernels

Output cost functions \( f \)

Under- and over-approximate by piecewise constant functions \( f_{\epsilon^-} \) and \( f_{\epsilon^+} \)

\( \sim \) reachability timed game in small regions
Approximation of the optimal cost

Proof

- **semi-unfolding** of region automaton (seen as a timed game)
- compute **exact** optimal cost in tree-like parts [LMM02, ABM04]
- compute **approximate** optimal cost in kernels

Under- and over-approximate by piecewise constant functions $f_{\epsilon}^-$ and $f_{\epsilon}^+$

Output cost functions $f$

$\sim$ reachability timed game in small regions
Outline of the talk
Conclusions and future directions

**Priced timed automata and games**

- convenient for modelling resources;
- **1-player** setting remains **tractable** (sort of);
- **2-player** setting **undecidable**, but **approximable**.
- **approximation algorithms** are a convenient **trade-off**.

**Future work**

- improve approximation technique (in terms of complexity);
- extend results to whole class of priced timed games;
- average energy and energy constraints;
- robust analysis of priced timed games;
- develop a tool.
Conclusions and future directions

**Priced timed automata and games**
- convenient for modelling resources;
- 1-player setting remains tractable (sort of);
- 2-player setting undecidable, but approximable.
- approximation algorithms are a convenient trade-off.

**Future work**
- improve approximation technique (in terms of complexity);
- extend results to whole class of priced timed games;
- average energy and energy constraints;
- robust analysis of priced timed games;
- develop a tool.
MOVSEP 2016

- 12th Summer School MOVSEP
- Genoa, Italy
- 27 June - 1 July
## MOVEP 2016

- 12th Summer School MOVEP
- Genoa, Italy
- 27 June - 1 July

## FORMATS 2016

- 14th Int. Conf. FORMATS
- colocated with CONCUR and QEST
- Quebec City, Canada