A journey through negatively-weighted timed games: undecidability, decidability, approximability

Benjamin Monmege, Aix-Marseille Université

WATA 2018, Leipzig
Motivation: quantitative aspects of real-time synthesis

$$\text{Environment} \parallel \text{Controller} \Rightarrow \text{Spec}$$
Motivation: quantitative aspects of real-time synthesis

\[
\text{Environment} \parallel \text{Controller??} \models \text{Spec}
\]

Real-time requirements/environment \(\implies\) real-time controller
Motivation: quantitative aspects of real-time synthesis

| Environment | Controller?? | = | Spec |

Real-time requirements/environment $\implies$ real-time controller

Among all valid controllers, choose a cheap/efficient one
Motivation: quantitative aspects of real-time synthesis

Environment $\parallel$ Controller?? $\models$ Spec

Two-player game

Real-time requirements/environment $\implies$ real-time controller

Among all valid controllers, choose a cheap/efficient one
Motivation: quantitative aspects of real-time synthesis

\[
\begin{array}{c|c|c}
\text{Environment} & \parallel & \text{Controller} \?? \\
\hline
\Rightarrow & & \text{Spec} \\
\end{array}
\]

Two-player game

Real-time requirements/environment \(\Rightarrow\) real-time controller

Two-player \textit{timed} game

Among all valid controllers, choose a cheap/efficient one
Motivation: quantitative aspects of real-time synthesis

Environment $\parallel$ Controller?? $\models$ Spec

Two-player game

Real-time requirements/environment $\rightarrow$ real-time controller

Two-player timed game

Among all valid controllers, choose a cheap/efficient one

Two-player weighted timed game
Motivation: quantitative aspects of real-time synthesis

Environment \parallel Controller?? \models Spec

Two-player game

Real-time requirements/environment \implies real-time controller

Two-player timed game

Among all valid controllers, choose a cheap/efficient one

Two-player weighted timed game

Additional difficulty: negative weights

\implies to model production/consumption of resources
Modelling via weighted timed games

Peak-hour 🌞

15 €/kWh
rate: total power × 15 €/h

Offpeak-hour 🌕

12 €/kWh
total power × 12 €/h

states to record which device is on/off: computation of the total power
Modelling via weighted timed games

Peak-hour 🌞

15 c€/kWh
rate: total power × 15 c€/h

Offpeak-hour 🌙

12 c€/kWh
rate: total power × 12 c€/h

states to record which device is on/off: computation of the total power

Power consumption:

▶ 100W (1.5 c€/h in peak-hour, 1.2 c€/h in offpeak-hour)

▶ 2500W (37.5 c€/h in peak-hour, 30 c€/h in offpeak-hour)

▶ 2000W (24 c€/h in offpeak-hour)
Modelling via weighted timed games

Peak-hour [Sun]  
15 c€/kWh  
rate: total power × 15 c€/h

Offpeak-hour [Moon]  
12 c€/kWh  
rate: total power × 12 c€/h

Solar panels  
Reselling: 20 c€/kWh  
−0.5 × 20 c€/h

*states to record which device is on/off: computation of the total power*
Modelling via weighted timed games

Peak-hour 🌞 15 c€/kWh
rate: total power × 15 c€/h

Offpeak-hour 🌙 12 c€/kWh
total power × 12 c€/h

Solar panels
Reselling: 20 c€/kWh
−0.5 × 20 c€/h

*states* to record which device is on/off: computation of the total power

**Environment:** user profile, weather profile 🌞 / ☁️

**Controller:** chooses contract (discrete cost for the monthly subscription) and exact consumption (what, when...)

Modelling via weighted timed games

Peak-hour  
15 c€/kWh  
rate: total power × 15 c€/h

Offpeak-hour  
12 c€/kWh  
total power × 12 c€/h

Solar panels  
Reselling: 20 c€/kWh  
−0.5 × 20 c€/h

states to record which device is on/off: computation of the total power

Environment: user profile, weather profile ☀️ / ☁️
Controller: chooses contract (discrete cost for the monthly subscription) and exact consumption (what, when...)

Goal: optimise the energy consumption based on the cost
Modelling via weighted timed games

<table>
<thead>
<tr>
<th>State</th>
<th>Price (c€/kWh)</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak-hour</td>
<td>15</td>
<td>total power × 15 c€/h</td>
</tr>
<tr>
<td>Offpeak-hour</td>
<td>12</td>
<td>total power × 12 c€/h</td>
</tr>
<tr>
<td>Solar panels</td>
<td>20</td>
<td>−0.5 × 20 c€/h</td>
</tr>
</tbody>
</table>

states to record which device is on/off: computation of the total power

Environment: user profile, weather profile ☀️ / ⛅️
Controller: chooses contract (discrete cost for the monthly subscription) and exact consumption (what, when...)

Goal: optimise the energy consumption based on the cost

Solution 1: discretisation of time, resolution via a weighted game
Solution 2: thin time behaviours, resolution via a weighted timed game
Weighted games

Weighted automaton with vertices partition between 2 players + reachability objective

Min = ∘, Max = □
Weighted games

Weighted automaton with vertices partition between 2 players + reachability objective

Weight of a path:
- \(+\infty\) if not reached
- total weight until otherwise

\[\text{Min} = \bigcirc, \text{Max} = \square\]
Weighted games

Weighted automaton with vertices partition between 2 players
+ reachability objective

Weight of a path: 
\[ \begin{cases} + \infty & \text{if not reached} \\ \text{total weight until otherwise} & \end{cases} \]

Min = ○, Max = □
Weighted games

Weighted automaton with vertices partition between 2 players + reachability objective

Weight of a path: \{ \begin{array}{ll} +\infty & \text{if not reached} \\ \text{total weight until otherwise} & \end{array} \}

Benjamin Monmege (Aix-Marseille Université) Min = ⊘, Max = □
Weighted games

Weighted automaton with vertices partition between 2 players + reachability objective

Weight of a path: 
\[ \{ + \infty \text{ if not reached} \text{ otherwise} \]
Weighted games

Weighted automaton with vertices partition between 2 players + reachability objective

Weight of a path: 

\[ \{+\infty \text{ if not reached}\} \]

Benjamin Monmege (Aix-Marseille Université)
Weighted games

Weighted automaton with vertices partition between 2 players + reachability objective

Weight of a path: \[
\begin{cases}
+\infty & \text{if } \checkmark \text{ not reached} \\
\text{total weight until } \checkmark & \text{otherwise}
\end{cases}
\]
Weighted timed games

Timed automaton with state partition between 2 players + reachability objective

Weight of an execution:
- \( \{ \infty \} \) if not reached
- Total weight until otherwise

Benjamin Monmege (Aix-Marseille Université)
Weighted timed games

Timed automaton with state partition between 2 players
+ reachability objective
+ linear rates on states
+ discrete weights on transitions

Weight of an execution:
\[
\begin{cases}
+ \infty & \text{if not reached} \\
\text{total weight until otherwise}
\end{cases}
\]
Weighted timed games

Timed automaton with state partition between 2 players
+ reachability objective
+ linear rates on states
+ discrete weights on transitions

Weight of an execution:
\[ \begin{cases} + \infty & \text{if not reached} \\ \text{total weight} & \text{otherwise} \end{cases} \]
Weighted timed games

Timed automaton
with state partition between
2 players
+ reachability objective
+ linear rates on states
+ discrete weights on transitions

Weight of an execution:
\{ + \infty \text{ if not reached}
\text{total weight until otherwise}\}

Benjamin Monmege (Aix-Marseille Université)

Min = \bigcirc, Max = \square
Weighted timed games

Timed automaton with state partition between 2 players

+ reachability objective
+ linear rates on states
+ discrete weights on transitions

Weight of an execution:

\[\{\begin{array}{ll}
\infty & \text{if not reached} \\
\text{total weight until } & \text{otherwise}
\end{array}\]
Weighted timed games

Timed automaton with state partition between 2 players
+ reachability objective
+ linear rates on states
+ discrete weights on transitions

Weight of an execution:

\[
\{(s_1, 0) \xrightarrow{0.4, \longrightarrow} (s_4, 0.4) \xrightarrow{0.6, \longrightarrow} (s_5, 0) \xrightarrow{1.5, \leftarrow} (s_4, 0) \xrightarrow{1.1, \rightarrow} (s_5, 0) \xrightarrow{2, \rightarrow} (\checkmark, 2)\}
\]

\[
1\times0.4+1 = 1.4
\]
\[
-3\times0.6+0 = -1.8
\]
\[
+1\times1.5+0 = 1.5
\]
\[
-3\times1.1+0 = -3.3
\]
\[
+1\times2+2 = 3
\]

\[
\text{total weight until } \checkmark, 2 = 1.8
\]

Benjamin Monmege (Aix-Marseille Université)
Weighted timed games

Timed automaton with state partition between 2 players
+ reachability objective
+ linear rates on states
+ discrete weights on transitions

```
Weight of an execution: {+
  +∞ if ✓ not reached
  total weight until ✓ otherwise
```

Benjamin Monmege (Aix-Marseille Université)
Strategies and objectives

Strategy for a player: map finite executions to a delay and a transition
Strategies and objectives

Strategy for a player: map finite executions to a delay and a transition

Objective of player $\bigcirc$: reach $\checkmark$ and minimise the weight
Objective of player $\square$: avoid $\checkmark$ or, if not possible, maximise the weight
Strategies and objectives

Strategy for a player: map finite executions to a delay and a transition

Objective of player ⊙: reach ✓ and minimise the weight
Objective of player : avoid ✓ or, if not possible, maximise the weight

Main object of interest:
\[ \text{Val}(s)(\nu) = \inf_{\sigma_{\text{Min}} \in \text{Strat}_{\text{Min}}} \sup_{\sigma_{\text{Max}} \in \text{Strat}_{\text{Max}}} \text{Weight}(\text{Exec}(s, \nu, \sigma_{\text{Min}}, \sigma_{\text{Max}})) \in \mathbb{R} \]

What weight can players guarantee? Following which strategies?

Benjamin Monmege (Aix-Marseille Université)
Part I: Weighted games
State of the art: weighted games (shortest-path objective)

\[ F_{\leq K} \checkmark \]: \exists \text{ a strategy in the weighted game for player } \bigcirc \text{ reaching } \checkmark \text{ with a cost } \leq K? \]

- one-player: shortest path in a weighted graph... polynomial algo.
- two players, non-negative weights only: polynomial algo. "Dijkstra algorithm on 2 players games" (Khachiyan et al., 2008)
State of the art: weighted games (shortest-path objective)

\[ F_{\leq K} \vdash: \exists \text{ a strategy in the weighted game for player } \bigcirc \text{ reaching } \checkmark \text{ with a cost } \leq K? \]

- one-player: shortest path in a weighted graph... polynomial algo.
- two players, non-negative weights only: polynomial algo.
  "Dijkstra algorithm on 2 players games" (Khachiyan et al., 2008)
- two players, arbitrary weights?

![Diagram of a weighted game with nodes and edges labeled with costs. The diagram shows a square node, a circle node, and transitions with labels -1, 0, -W, 0, and NO.]
State of the art: weighted games (shortest-path objective)

\[ F_{\leq K} \vdash : \exists \text{ a strategy in the weighted game for player } \bigcirc \text{ reaching } \checkmark \text{ with a cost } \leq K? \]

- one-player: shortest path in a weighted graph... polynomial algo.
- two players, non-negative weights only: polynomial algo. "Dijkstra algorithm on 2 players games" (Khachiyan et al., 2008)
- two players, arbitrary weights?

\[ \bigcirc \text{ needs memory!} \]

Value \(-\infty\): detection is as hard as solving parity games (\(\text{NP} \cap \text{co-NP}\))
Pseudo-polynomial algorithm to solve weighted games

Joint work with Thomas Brihaye, Gilles Geeraerts and Axel Haddad (Brihaye et al., 2016)

Value iteration algorithm: compute $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(x)_v = \begin{cases} 
\min_{e=(v,a,v') \in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Min}} \\
\max_{e=(v,a,v') \in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Max}} 
\end{cases}$$

[Diagram of a game tree with states and transitions labeled with values and actions.]

**Theorem:** We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players: $\text{Min} = \#$, $\text{Max} = 2^{9/34}$. 

Benjamin Monmege (Aix-Marseille Université)
Pseudo-polynomial algorithm to solve weighted games

Joint work with Thomas Brihaye, Gilles Geeraerts and Axel Haddad \cite{Brihaye2016}

Value iteration algorithm: compute $F^i(\infty)$...

$$F(x)_v = \begin{cases} \min_{e=(v,a,v')\in E} \left(\text{Weight}(e) + x_{v'}\right) & \text{if } v \in V_{\text{Min}} \\ \max_{e=(v,a,v')\in E} \left(\text{Weight}(e) + x_{v'}\right) & \text{if } v \in V_{\text{Max}} \end{cases}$$

horizon 0: $+\infty$ $+\infty$
horizon 1: $+\infty$ 0
Pseudo-polynomial algorithm to solve weighted games

Joint work with Thomas Brihaye, Gilles Geeraerts and Axel Haddad (Brihaye et al., 2016)

Value iteration algorithm: compute $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(x)_v = \begin{cases} 
\min_{e=(v,a,v') \in E} (\text{Weight}(e) + x_{v'}) & \text{if } v \in V_{\text{Min}} \\
\max_{e=(v,a,v') \in E} (\text{Weight}(e) + x_{v'}) & \text{if } v \in V_{\text{Max}} 
\end{cases}$$

Theorem: We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players: may require (pseudo-polynomial) memory to play optimally (but has counter strategies), has optimal memoryless strategy.

Benjamin Monmege (Aix-Marseille Université)

$\text{Min} = \#$, $\text{Max} = 2^{9/34}$
Pseudo-polynomial algorithm to solve weighted games

Joint work with Thomas Brihaye, Gilles Geeraerts and Axel Haddad (Brihaye et al., 2016)

Value iteration algorithm: compute \( \mathcal{F}^i(\infty) \)...

\[
\mathcal{F}(x)_v = \begin{cases} 
\min_{e=(v,a,v') \in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Min}} \\
\max_{e=(v,a,v') \in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Max}} 
\end{cases}
\]

Theorem: We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players.

May require (pseudo-polynomial) memory to play optimally (but has counter strategies).

Optimal memoryless strategy.

Benjamin Monmege (Aix-Marseille Université)

\( V_{\text{Min}} = \#, V_{\text{Max}} = 2 \)

\( W = \frac{34}{9} \)
Pseudo-polynomial algorithm to solve weighted games

Joint work with Thomas Brihaye, Gilles Geeraerts and Axel Haddad (Brihaye et al., 2016)

Value iteration algorithm: compute $\mathcal{F}^i(+\infty)$...

$$
\mathcal{F}(x)_v = \begin{cases} 
\min_{e=(v,a,v') \in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Min}} \\
\max_{e=(v,a,v') \in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Max}}
\end{cases}
$$

---

Theorem:
We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players: may require (pseudo-polynomial) memory to play optimally (but has counter strategies), 2 has optimal memoryless strategy.
Pseudo-polynomial algorithm to solve weighted games

Joint work with Thomas Brihaye, Gilles Geeraerts and Axel Haddad (Brihaye et al., 2016)

Value iteration algorithm: compute $F^i(+\infty)$...

$F(x)_v = \begin{cases} 
\min_{e=(v,a,v') \in E} (\text{Weight}(e) + x_{v'}) & \text{if } v \in V_{\text{Min}} \\
\max_{e=(v,a,v') \in E} (\text{Weight}(e) + x_{v'}) & \text{if } v \in V_{\text{Max}} 
\end{cases}$

---

Theorem: We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players:

- $\text{Min} = \frac{9}{34}$
- $\text{Max} = 2$

optimal memoryless strategy.

---

Benjamin Monmege (Aix-Marseille Université)

```
horizon 0: +\infty +\infty
horizon 1: +\infty 0
horizon 2: -1 0
horizon 3: -1 -1
horizon 4: -2 -1
... ...
horizon 2W + 1: -W -W
```
Pseudo-polynomial algorithm to solve weighted games

Joint work with Thomas Brihaye, Gilles Geeraerts and Axel Haddad (Brihaye et al., 2016)

Value iteration algorithm: compute $\mathcal{F}^i(\infty)$...

$$\mathcal{F}(x)_v = \begin{cases} 
\min_{e=(v,a,v') \in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Min}} \\
\max_{e=(v,a,v') \in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Max}}
\end{cases}$$

Theorem: We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players:

$\text{Min} = \bigcirc$, $\text{Max} = \square$

Benjamin Monmege (Aix-Marseille Université)
Pseudo-polynomial algorithm to solve weighted games

Joint work with Thomas Brihaye, Gilles Geeraerts and Axel Haddad (Brihaye et al., 2016)

Value iteration algorithm: compute $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(x)_v = \begin{cases} \min_{e=(v,a,v')\in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Min}} \\ \max_{e=(v,a,v')\in E} \left( \text{Weight}(e) + x_{v'} \right) & \text{if } v \in V_{\text{Max}} \end{cases}$$

Theorem:

We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players: ○ may require (pseudo-polynomial) memory to play optimally (but has counter strategies), □ has optimal memoryless strategy.
Large polynomial fragment: divergent weighted games

Joint work with Damien Busatto-Gaston and Pierre-Alain Reynier (Busatto-Gaston et al., 2017)

Divergence property (in the underlying graph):
Every cycle has total weight either $\leq -1$ or $\geq 1$

Theorem: We can compute in polynomial time the value of a divergent weighted game, as well as optimal strategies for both players.

Theorem: Deciding if a weighted game is divergent is in PTIME.
Large polynomial fragment: divergent weighted games

Joint work with Damien Busatto-Gaston and Pierre-Alain Reynier (Busatto-Gaston et al., 2017)

Divergence property (in the underlying graph):
Every cycle has total weight either $\leq -1$ or $\geq 1$

**Theorem:**
We can compute in polynomial time the value of a divergent weighted game, as well as optimal strategies for both players.
Large polynomial fragment: divergent weighted games

Joint work with Damien Busatto-Gaston and Pierre-Alain Reynier (Busatto-Gaston et al., 2017)

Divergence property (in the underlying graph):
**Every cycle has total weight either \( \leq -1 \) or \( \geq 1 \)**

**Theorem:**
We can compute in polynomial time the value of a divergent weighted game, as well as optimal strategies for both players.

**Theorem:**
Deciding if a weighted game is divergent is in PTIME.

Benjamin Monmege (Aix-Marseille Université)
Divergent weighted games analysis

\[ p \geq 1 \]
\[ -q \leq -1 \]

Benjamin Monmege (Aix-Marseille Université)
Divergent weighted games analysis

characterisation: All the simple cycles in a SCC have the same sign

\[ p \geq 1 \]
\[ -q \leq -1 \]

\[ p \geq 1 \]
\[ -q \leq -1 \]
Divergent weighted games analysis

characterisation: All the simple cycles in a SCC have the same sign

divergence property

class decision

value computation
Value computation in a divergent weighted game

- Detect and remove $+\infty$ vertices (outside of the attractor of player $\textcircled{o}$ toward $\checkmark$)
Value computation in a divergent weighted game

- Detect and remove $+\infty$ vertices (outside of the attractor of player $\bigcirc$ toward $\checkmark$)
- SCC decomposition
- Value computation SCC by SCC, bottom-up
Value computation in a divergent weighted game

- Detect and remove $+\infty$ vertices (outside of the attractor of player $\bigcirc$ toward $\checkmark$)
- SCC decomposition
- Value computation SCC by SCC, bottom-up

**positive SCC**

- The "value iteration" algorithm converges in linear time
Value computation in a divergent weighted game

- Detect and remove $+\infty$ vertices (outside of the attractor of player ○ toward ✓)
- SCC decomposition
- Value computation SCC by SCC, bottom-up

Positive SCC

- The "value iteration" algorithm converges in linear time

Negative SCC

- Outside of the attractor of player □ toward ✓ $\Rightarrow -\infty$
- The "value iteration" algorithm converges in linear time with initialisation at $-\infty$
Example

\[ \text{Min} = \bigcirc, \text{Max} = \square \]
Example

\[ v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_8 \rightarrow v_f \]

\[ v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_8 \rightarrow v_f \]

\[ v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_8 \rightarrow v_f \]

\[ v_7 \rightarrow v_8 \rightarrow v_f \]

\[ v_9 \rightarrow v_8 \rightarrow v_f \]

Min = \( \bigcirc \), Max = \( \square \)
Example

\[
\begin{aligned}
&v_1 \quad v_2 \quad v_3 \\
&v_4 \quad v_5 \quad v_6 \\
&v_7 \quad v_8 \quad v_9 \\
&v_f
\end{aligned}
\]

\[
\begin{aligned}
&-1 \\
&-1 \\
&-1 \\
&-1 \\
&-1 \\
&-1 \\
&-1 \\
&-1 \\
&-1 \\
&-1 \\
&1 \\
&1 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \\
&2 \

Example
Example
Example

\[
\begin{align*}
    v_1 & \rightarrow v_2 & \rightarrow v_3 & \rightarrow v_4 \\
    v_5 & \rightarrow v_6 & \rightarrow v_7 & \rightarrow v_8 \\
    v_9 & \rightarrow v_f
\end{align*}
\]
Example
Example

\begin{itemize}
  \item \(v_1\) with label \(-\infty\)
  \item \(v_2\) with label \(-9\)
  \item \(v_3\) with label \(-\infty\)
  \item \(v_4\) with label \(+\infty\)
  \item \(v_5\) with label \(-1\)
  \item \(v_6\) with label \(1\)
  \item \(v_7\) with label \(+\infty\)
  \item \(v_8\) with label \(1\)
  \item \(v_9\) with label \(-1\)
  \item \(v_f\) with label \(\checkmark\)
\end{itemize}
Example
Part II : Weighted timed games
State of the art

$F_{\leq K}^\checkmark$: $\exists$ a strategy in the WTG (weighted timed game) for player $\bigcirc$ reaching $\checkmark$ with a cost $\leq K$?
State of the art

\[ F_{\leq K} \vDash : \exists \text{ a strategy in the WTG (weighted timed game) for player } \bigodot \text{ reaching } \checkmark \text{ with a cost } \leq K? \]

- **One-player case (Weighted timed automata):** optimal reachability problem is PSPACE-complete
  - Algorithm based on regions (Bouyer et al., 2004a, 2007);
  - and hardness shown for timed automata with at least 2 clocks (Fearnley and Jurdziński, 2013; Haase et al., 2012)
State of the art

$F_{\leq K} \checkmark$: $\exists$ a strategy in the WTG (weighted timed game) for player $\bigcirc$ reaching $\checkmark$ with a cost $\leq K$?

- One-player case (*Weighted timed automata*): optimal reachability problem is PSPACE-complete
  - Algorithm based on regions (*Bouyer et al., 2004a, 2007*);
  - and hardness shown for timed automata with at least 2 clocks (*Fearnley and Jurdziński, 2013; Haase et al., 2012*)

- 2-player WTGs: *undecidable* (*Brihaye et al., 2005; Bouyer et al., 2006a*), even with only non-negative weights and 3 clocks (only 2 clocks needed with arbitrary weights (*Brihaye et al., 2014*))
State of the art

\[ F_{\leq K}^\checkmark : \exists \text{ a strategy in the WTG (weighted timed game) for player } \bigcirc \text{ reaching } \checkmark \text{ with a cost } \leq K? \]

- **One-player case (Weighted timed automata):** optimal reachability problem is PSPACE-complete
  - Algorithm based on regions (Bouyer et al., 2004a, 2007);
  - and hardness shown for timed automata with at least 2 clocks (Fearnley and Jurdziński, 2013; Haase et al., 2012)

- **2-player WTGs:** undecidable (Brihaye et al., 2005; Bouyer et al., 2006a), even with only non-negative weights and 3 clocks (only 2 clocks needed with arbitrary weights (Brihaye et al., 2014))

- **WTGs with non-negative weights and strictly non-Zeno weight cycles:** 2-exponential algorithm (Bouyer et al., 2004b; Alur et al., 2004a)
State of the art

\( F_{\leq K} \): \exists \text{ a strategy in the WTG (weighted timed game) for player } \bigcirc \text{ reaching } \checkmark \text{ with a cost } \leq K \)

- **One-player case (Weighted timed automata):** optimal reachability problem is PSPACE-complete
  - Algorithm based on regions (Bouyer et al., 2004a, 2007);
  - and hardness shown for timed automata with at least 2 clocks (Fearnley and Jurdziński, 2013; Haase et al., 2012)

- **2-player WTGs:** undecidable (Brihaye et al., 2005; Bouyer et al., 2006a), even with only non-negative weights and 3 clocks (only 2 clocks needed with arbitrary weights (Brihaye et al., 2014))

- **WTGs with non-negative weights and strictly non-Zeno weight cycles:** 2-exponential algorithm (Bouyer et al., 2004b; Alur et al., 2004a)

- **One-clock WTGs with non-negative weights:** exponential algorithm (Bouyer et al., 2006b; Rutkowski, 2011; Hansen et al., 2013)
State of the art

\( F_{\leq K} \):  \exists \) a strategy in the WTG (weighted timed game) for player \( \bigcirc \) reaching \( \checkmark \) with a cost \( \leq K \)?

- **One-player case (Weighted timed automata)**: optimal reachability problem is PSPACE-complete
  - Algorithm based on regions (Bouyer et al., 2004a, 2007);
  - and hardness shown for timed automata with at least 2 clocks (Fearnley and Jurdziński, 2013; Haase et al., 2012)

- 2-player WTGs: **undecidable** (Brihaye et al., 2005; Bouyer et al., 2006a), even with only non-negative weights and 3 clocks (only 2 clocks needed with arbitrary weights (Brihaye et al., 2014))

- WTGs with **non-negative weights and strictly non-Zeno weight cycles**: 2-exponential algorithm (Bouyer et al., 2004b; Alur et al., 2004a)

- **One-clock** WTGs with **non-negative weights**: exponential algorithm (Bouyer et al., 2006b; Rutkowski, 2011; Hansen et al., 2013)

- Decidability results for WTGs with arbitrary weights?
One-player case: weighted timed automata

- Main tool: refinement of regions via corner point abstraction / $\varepsilon$-graph (Bouyer et al., 2004a, 2007)
One-clock Bi-Valued WTGs (1BWTGs)
One-clock Bi-Valued WTGs (1BWTGs)

Joint work with Thomas Brihaye, Gilles Geeraerts, Shankara Krishna Narayanan, Lakshmi Manasa and Ashutosh Trivedi (Brihaye et al., 2014)

**Assumption:** rates of locations $\{p^-, p^+\}$ included in $\{0, +d, -d\}$ ($d \in \mathbb{N}$) (no assumption on costs of transitions)

$$x < 1, x := 0, 0$$

$$x > 0$$
$$x := 0, 0$$

$$s_1$$

$$[x \leq 1]$$

$$s_2$$

$$x \leq 2, 0$$

$$s_3$$

$$x > 1, 1$$

$$s_4$$

$$x \leq 1, 1$$

$$s_5$$

$$x \geq 1, 2$$

$$s_6$$

$$x \geq 1$$
$$x := 0, 0$$

$$x := 0, 0$$

$$x \geq 1$$

$$[x \leq 2]$$

$$[x \leq 2]$$

Benjamin Monmege (Aix-Marseille Université)
One-clock Bi-Valued WTGs (1BWTGs)

Joint work with Thomas Brihaye, Gilles Geeraerts, Shankara Krishna Narayanan, Lakshmi Manasa and Ashutosh Trivedi (Brihaye et al., 2014)

Assumption: rates of locations \( \{p^-, p^+\} \) included in \( \{0, +d, -d\} \) (no assumption on costs of transitions)

\[
x < 1, x := 0, 0
\]
\[
x > 0, x := 0, 0
\]
\[
x > 1, 1 \leq x \leq 2, x := 0, 0
\]
\[
x > 1, 1 \leq x \leq 2
\]

regions: \( \{0\}, (0, 1), \{1\}, (1, 2), \{2\}, (2, +\infty) \)
regions refined with corner information:
\( \{0\}, (0, \eta), (1 - \eta, 1), \{1\}, (1, 1 + \eta), (2 - \eta, 2), \{2\}, (2, +\infty) \)
One-clock Bi-Valued WTGs (1BWTGs)

Joint work with Thomas Brihaye, Gilles Geeraerts, Shankara Krishna Narayanan, Lakshmi Manasa and Ashutosh Trivedi (Brihaye et al., 2014)

Assumption: rates of locations \( \{p^-, p^+\} \) included in \( \{0, +d, -d\} \) (\( d \in \mathbb{N} \)) (no assumption on costs of transitions)
One-clock Bi-Valued WTGs (1BWTGs)

Joint work with Thomas Brihaye, Gilles Geeraerts, Shankara Krishna Narayanan, Lakshmi Manasa and Ashutosh Trivedi (Brihaye et al., 2014)

**Assumption:** rates of locations \( \{p^-, p^+\} \) included in \( \{0, +d, -d\} \) (\( d \in \mathbb{N} \)) (no assumption on costs of transitions)
One-clock Bi-Valued WTGs (1BWTG)

Joint work with Thomas Brihaye, Gilles Geeraerts, Shankara Krishna Narayanan, Lakshmi Manasa and Ashutosh Trivedi (Brihaye et al., 2014)

Assumption: rates of locations \( \{p^-, p^+\} \) included in \( \{0, +d, -d\} \) \((d \in \mathbb{N})\) (no assumption on costs of transitions)

### Theorem:

Computation of the value \( \overline{\text{Val}}(s, \nu) \) of states of a 1BWTG and synthesis of \( \varepsilon \)-optimal strategies for \( \bigcirc \) in pseudo-polynomial time

- Only non-negative costs \( \implies \) polynomial time

Benjamin Monmege (Aix-Marseille Université)
Crucial tool: symmetrize the viewpoint

Value for player □:
\[ \overline{\text{Val}}(s, v) = \inf_{\sigma_{\text{Min}} \in \text{Strat}^{\text{Min}}} \sup_{\sigma_{\text{Max}} \in \text{Strat}^{\text{Max}}} \text{Weight(Exec}((s, v), \sigma_{\text{Min}}, \sigma_{\text{Max}})) \]

Value for player ◯:
\[ \underline{\text{Val}}(s, v) = \sup_{\sigma_{\text{Max}} \in \text{Strat}^{\text{Max}}} \inf_{\sigma_{\text{Min}} \in \text{Strat}^{\text{Min}}} \text{Weight(Exec}((s, v), \sigma_{\text{Min}}, \sigma_{\text{Max}})) \]

How to compare them? \[ \underline{\text{Val}}(s, v) \leq \overline{\text{Val}}(s, v) \]
Crucial tool: symmetrize the viewpoint

Value for player $\bigcirc$:
\[
\overline{\text{Val}}(s, \nu) = \inf_{\sigma_{\text{Min}} \in \text{Strat}^{\text{Min}}} \sup_{\sigma_{\text{Max}} \in \text{Strat}^{\text{Max}}} \text{Weight}(\text{Exec}((s, \nu), \sigma_{\text{Min}}, \sigma_{\text{Max}}))
\]

Value for player $\blacksquare$:
\[
\underline{\text{Val}}(s, \nu) = \sup_{\sigma_{\text{Max}} \in \text{Strat}^{\text{Max}}} \inf_{\sigma_{\text{Min}} \in \text{Strat}^{\text{Min}}} \text{Weight}(\text{Exec}((s, \nu), \sigma_{\text{Min}}, \sigma_{\text{Max}}))
\]

How to compare them? $\overline{\text{Val}}(s, \nu) \leq \underline{\text{Val}}(s, \nu)$

Theorem: Minmax theorem

- 1BPGs (and even all WTGs (Brihaye et al., 2015)) are determined, i.e.,
  \[
  \overline{\text{Val}}(s, \nu) = \underline{\text{Val}}(s, \nu)
  \]

- Synthesis of $\varepsilon$-optimal strategies for player $\blacksquare$ in pseudo-polynomial time (and polynomial in case of non-negative weights)
1BWTG: maximal fragment for corner-point abstraction

Generalisation by Engel Lefaucheux: two rates \( \{p^-, p^+\} \) included in \( \{0, +d, -c\} \ (d, c \in \mathbb{N}) \)

*In more general settings, players may need to play far from corners...*

▶ With 3 weights in \( \{-1, 0, +1\} \): value 1/2...

\[
\begin{align*}
x &\leq 1, x := 0 & x \leq 1, x := 0 \\
x &= 1 & x &= 1
\end{align*}
\]
1BWTG: maximal fragment for corner-point abstraction

Generalisation by Engel Lefaucheux: two rates \( \{ p^-, p^+ \} \) included in \( \{0, +d, -c\} \) (\( d, c \in \mathbb{N} \))

In more general settings, players may need to play far from corners...

- With 3 weights in \( \{-1, 0, +1\} \): value 1/2...

- With 2 weights in \( \{-1, 0, +1\} \) but 2 clocks: value 1/2...

- How to push further the resolution of WTGs?
One-clock WTG... Almost!
Related work: 1-clock, non-negative weights

(Hansen et al., 2013): strategy improvement algorithm
(Bouyer et al., 2006b; Rutkowski, 2011): iterative elimination of locations
  ► precomputation: polynomial-time cascade of simplification of 1-clock WTGs into simple 1-clock WTGs (SWTGs)
    ► clock bounded by 1, no guards/invariants, no resets
Related work: 1-clock, non-negative weights

(Hansen et al., 2013): strategy improvement algorithm
(Bouyer et al., 2006b; Rutkowski, 2011): iterative elimination of locations

- precomputation: polynomial-time cascade of simplification of 1-clock W TGs into simple 1-clock W TGs (SW TGs)
  - clock bounded by 1, no guards/invariants, no resets
- for SW TGs: compute value functions $\overline{\text{Val}}(\ell, x)$.
SWTGs with arbitrary weights

Joint work with Thomas Brihaye, Gilles Geeraerts, Axel Haddad and Engel Lefaucheux (Brihaye et al., 2015)
SWTGs with arbitrary weights

Joint work with Thomas Brihaye, Gilles Geeraerts, Axel Haddad and Engel Lefaucheux (Brihaye et al., 2015)

\[ Val(\ell_4, x) = \sup_{0 \leq t \leq 1 - x} 3t - 7 = 3(1 - x) - 7 = -3x - 4 \]
SWTGs with arbitrary weights

Joint work with Thomas Brihaye, Gilles Geeraerts, Axel Haddad and Engel Lefaucheux (Brihaye et al., 2015)

\[
\begin{align*}
\ell_6 & : -12 \\
\ell_5 & : 8 \\
\ell_1 & : -2 \\
\ell_2 & : -14 \\
\ell_3 & : 4 \\
\ell_4 & : 3 \\
\ell_7 & : -16 \\
\end{align*}
\]

Val(\(\ell_4, x\)) = \(-3x - 4\), \hspace{1cm} Val(\(\ell_7, x\)) = \(-16(1 - x)\)

Benjamin Monmege (Aix-Marseille Université)
SWTGs with arbitrary weights

Joint work with Thomas Brihaye, Gilles Geeraerts, Axel Haddad and Engel Lefaucheux (Brihaye et al., 2015)

\[
\begin{align*}
\text{Val}(\ell_4, x) &= -3x - 4, \\
\text{Val}(\ell_7, x) &= -16(1 - x), \\
\text{Val}(\ell_3, x) &= \inf_{0 \leq t \leq 1-x} [4t + \min(-3(x + t) - 4, 6 - 16(1 - (x + t)))] = \\
&\min(-3x - 4, 16x - 10)
\end{align*}
\]
Recursive elimination of states

- Player $\bigcirc$ prefers to stay as long as possible in locations with **minimal rate**: add a final location allowing him to stay until the end, and make the location urgent.
Recursive elimination of states

- Player ◯ prefers to stay as long as possible in locations with **minimal rate**: add a final location allowing him to stay until the end, and make the location urgent
- Player □ prefers to leave as soon as possible in locations with **minimal rate**: make the location urgent

Theorem: For every SWTG, all value functions are piecewise affine, with at most an exponential number of cutpoints (in number of locations).
Recursive elimination of states

- Player ◯ prefers to stay as long as possible in locations with **minimal rate**: add a final location allowing him to stay until the end, and make the location urgent.
- Player □ prefers to leave as soon as possible in locations with **minimal rate**: make the location urgent.

**Theorem:**
For every SWTG, all value functions are piecewise affine, with at most an exponential number of cutpoints (in number of locations).

For general 1-clock WTGs?
- removing guards and invariants: previously used techniques work!
- removing resets: previously, bound the number of resets...
Solving SWTGs with arbitrary weights

Val($\ell_3, x$) = 0 \rightarrow -6 \rightarrow -5.5 \rightarrow -7

Val($\ell_4, x$) = 0 \rightarrow -4 \rightarrow -7

Val($\ell_5, x$) = 0 \rightarrow -16

Val($\ell_2, x$) = 0 \rightarrow -2 \rightarrow -6 \rightarrow -5.5

Val($\ell_1, x$) = 0 \rightarrow -0.2 \rightarrow -2

Val($\ell_7, x$) = 0 \rightarrow -14

Benjamin Monmege (Aix-Marseille Université)

Min = ○, Max = □
Bounding the number of resets needed is not possible

\[ x = 1, x := 0 \]

\[ x \leq 1 \]

\[ W \]

Best we can do: exponential time algorithm for reset-acyclic 1-clock WTGs with arbitrary weights
Bounding the number of resets needed is not possible

Player $\bigcirc$ can guarantee (i.e., ensure to be below) value $\varepsilon$ for all $\varepsilon > 0$...
Bounding the number of resets needed is not possible

\[ x = 1, x := 0 \]

Player \( \bigcirc \) can guarantee (i.e., ensure to be below) value \( \varepsilon \) for all \( \varepsilon > 0 \)... 

... but cannot obtain 0: hence, no optimal strategy...
Bounding the number of resets needed is not possible

Player $\bigcirc$ can guarantee (i.e., ensure to be below) value $\varepsilon$ for all $\varepsilon > 0$...

... but cannot obtain 0: hence, no optimal strategy...

... moreover, to obtain $\varepsilon$, $\bigcirc$ needs to loop at least $W + \lceil 1/\log \varepsilon \rceil$ times before reaching $\checkmark$!
Bounding the number of resets needed is not possible

![Graph](image)

Player $\bigcirc$ can guarantee (i.e., ensure to be below) value $\varepsilon$ for all $\varepsilon > 0$...

... but cannot obtain 0: hence, no optimal strategy...

... moreover, to obtain $\varepsilon$, $\bigcirc$ needs to loop at least $W + \lceil 1/\log \varepsilon \rceil$ times before reaching $\checkmark$!

**Best we can do:** exponential time algorithm for reset-acyclic 1-clock WTGs with arbitrary weights

Benjamin Monmege (Aix-Marseille Université) Min $= \bigcirc$, Max $= \Box$
Finally several clocks...
More than one clock?

**non-negative weights and strictly non-Zeno-cost cycles:**

2-exponential algorithm (Bouyer et al., 2004c; Alur et al., 2004b)

Value iteration algorithm: compute $\mathcal{F}^i(\infty)$...

\[
\mathcal{F}(x)_{(s,\nu)} = \begin{cases} 
\sup_{(s,\nu) \xrightarrow{d,t} (s',\nu')} (d \times \text{Weight}(s) + \text{Weight}(t) + x(s',\nu')) & \text{if } s \in S_{\text{Max}} \\
\inf_{(s,\nu) \xrightarrow{d,t} (s',\nu')} (d \times \text{Weight}(s) + \text{Weight}(t) + x(s',\nu')) & \text{if } s \in S_{\text{Min}}
\end{cases}
\]

Stabilises after a number of iterations at most exponential in the size of the game (because of the number of regions)
Extension to negative weights

Joint work with Damien Busatto-Gaston and Pierre-Alain Reynier (Busatto-Gaston et al., 2017)

Divergence property (of the underlying timed automaton):
Every execution following a cycle of the region automaton has a total weight either \(\leq -1\) or \(\geq 1\)
Extension to negative weights

Joint work with Damien Busatto-Gaston and Pierre-Alain Reynier (Busatto-Gaston et al., 2017)

Divergence property (of the underlying timed automaton):
Every execution following a cycle of the region automaton has a total weight either $\leq -1$ or $\geq 1$

**Theorem:**
The value problem on divergent weighted timed games is in $2^{\text{EXP}}$, and is $\text{EXP}$-hard.
Extension to negative weights

Joint work with Damien Busatto-Gaston and Pierre-Alain Reynier (Busatto-Gaston et al., 2017)

Divergence property (of the underlying timed automaton):
Every execution following a cycle of the region automaton has a total weight either $\leq -1$ or $\geq 1$

**Theorem:**
The value problem on divergent weighted timed games is in $2$-EXP, and is EXP-hard.

**Theorem:**
Deciding if a weighted timed game is divergent is PSPACE-complete.
Weighted timed games analysis

\[ \geq 1 \]

\[ \leq -1 \]

\[ \text{divergence property} \]

\[ \text{characterisation:} \]

\[ \text{Min} = \bigcirc, \text{Max} = \square \]
Weighted timed games analysis

\[ \geq 1 \]

\[ \leq -1 \]

divergence property

characterisation:

Min = ⬤, Max = □
Weighted timed games analysis

\[ \geq 1 \]

\[ \leq -1 \]

\[ \text{divergence property} \]

\[ \text{characterisation:} \]

\[ \geq 1 \]

\[ \leq -1 \]

Min = ○, Max = □
Weighted timed games analysis

divergence property

characterisation: All the simple cycles in a SCC have the same sign
Weighted timed games analysis

Divergence property:

Characterisation: All the simple cycles in a SCC have the same sign

Class decision

Value computation
Value computation in divergent weighted timed games

- Remove $+\infty$ states
- SCC decomposition
- Value computation SCC after SCC, bottom-up
Value computation in divergent weighted timed games

- Remove $+\infty$ states
- SCC decomposition
- Value computation SCC after SCC, bottom-up

**positive SCC**

- weighted timed games with **non-negative weights and strictly non-Zeno-cost cycles** (Bouyer et al., 2004c; Alur et al., 2004b)
- The iterative algorithm converges in a number of steps linear with the region automaton’s size
Value computation in divergent weighted timed games

- Remove $+\infty$ states
- SCC decomposition
- Value computation SCC after SCC, bottom-up

**positive SCC**

- weighted timed games with **non-negative weights and strictly non-Zeno-cost cycles** (Bouyer et al., 2004c; Alur et al., 2004b)
- The iterative algorithm converges in a number of steps linear with the region automaton’s size

**negative SCC**

- Outside of the attractor of player $\square$ toward $\checkmark \Rightarrow -\infty$
- The iterative algorithm converges on the other states in a number of steps linear with the region automaton’s size, with $-\infty$ initialisation
What about cycles of weight $= 0$?

- Adding cycles of weight $= 0$ to divergent WTG $\implies$ **Undecidable!**

(Bouyer et al., 2015)
What about cycles of weight $= 0$?

- Adding cycles of weight $= 0$ to divergent WTG $\implies$ **Undecidable**!
- Already with only non-negative weights (Bouyer et al., 2015): but possible to approximate the value (with elementary complexity)...

Min $= \bigcirc$, Max $= \blacksquare$
Extension in the negative case?

Ongoing work with Damien Busatto-Gaston and Pierre-Alain Reynier

Almost-divergent WTG: every SCC of the region automaton is

either ($\geq 1$ or $= 0$), or ($\leq -1$ or $= 0$)
Extension in the negative case?

Ongoing work with Damien Busatto-Gaston and Pierre-Alain Reynier

Almost-divergent WTG: every SCC of the region automaton is

either $(\geq 1 \text{ or } 0)$, or $(\leq -1 \text{ or } 0)$

Theorem: Approximation is decidable (with elementary complexity) for almost-divergent WTGs: (semi-)symbolic algorithm that does not rely on an a-priori discretisation of the regions with a fixed granularity $\frac{1}{N}$ (as in (Bouyer et al., 2015))

$\triangleright$ circumvent the need for an SCC decomposition?

$\text{Min} = \bigcirc$, $\text{Max} = \square$
Extension in the negative case?

Ongoing work with Damien Busatto-Gaston and Pierre-Alain Reynier

Almost-divergent WTG: every SCC of the region automaton is
either ($\geq 1$ or $= 0$), or ($\leq -1$ or $= 0$)

Theorem:
Approximation is decidable (with elementary complexity) for almost-divergent WTGs: (semi-)symbolic algorithm that does not rely on an a-priori discretisation of the regions with a fixed granularity $1/N$ (as in (Bouyer et al., 2015))

▸ circumvent the need for an SCC decomposition?

Benjamin Monmege (Aix-Marseille Université)
Conclusion

1BWTG

poly / pseudo-poly

(+) (-)

Thank you!
Conclusion

1WTG reset-acyclic
exp / exp
poly-hard

1BWTG
poly (+) / pseudo-poly (-)

Thank you!
Conclusion

1WTG reset-acyclic
exp / exp
poly-hard

1BWTG
poly (+) / pseudo-poly (-)

divergent WTG
2-exp / 2-exp
exp-hard

Benjamin Monmege (Aix-Marseille Université)
Min = ○, Max = □
Conclusion

1WTG reset-acyclic
exp / exp
poly-hard

1BWTG
poly (⁺)
/pseudo-poly (⁻)

almost-divergent WTG
approx / approx
elementary complexity

divergent WTG
2-exp / 2-exp
exp-hard

Thank you!

Benjamin Monmege (Aix-Marseille Université)
Conclusion

WTG
undec / undec
\( \geq 3 \text{ clocks} / \geq 2 \text{ clocks} \)

1WTG reset-acyclic
exp / exp
poly-hard

1BWTG
poly (+)
pseudo-poly (-)

almost-divergent WTG
approx / approx
elementary complexity

divergent WTG
2-exp / 2-exp
exp-hard

\[ \text{Min} = \circ, \text{Max} = \square \]
Conclusion

WTG
undec / undec
\( \geq 3 \) clocks / \( \geq 2 \) clocks

1WTG reset-acyclic
exp / exp
poly-hard

1BWTG
poly (+)
pseudo-poly (-)

almost-divergent WTG
approx / approx
elementary complexity

divergent WTG
2-exp / 2-exp
exp-hard

1WTG?

Min = \( \bigcirc \), Max = \( \square \)
Conclusion

<table>
<thead>
<tr>
<th>WTG</th>
<th>1WTG reset-acyclic</th>
<th>1BWTG</th>
<th>divergent WTG</th>
</tr>
</thead>
<tbody>
<tr>
<td>undec</td>
<td>exp / exp</td>
<td>poly</td>
<td>2-exp</td>
</tr>
<tr>
<td>⩾ 3 clocks</td>
<td>poly-hard</td>
<td>(+)</td>
<td>exp-hard</td>
</tr>
<tr>
<td>⩾ 2 clocks</td>
<td></td>
<td>(-)</td>
<td></td>
</tr>
<tr>
<td>approx</td>
<td></td>
<td>approx</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>elementary complexity</td>
</tr>
</tbody>
</table>

Thank you!

Benjamin Monmege (Aix-Marseille Université)
Conclusion

WTG
- undec / undec
- $\geq 3$ clocks / $\geq 2$ clocks

1WTG reset-acyclic
- exp / exp
- poly-hard

1BWTG
- poly (⁺)
- pseudo-poly (⁻)

almost-divergent WTG
- approx / approx
- elementary complexity

divergent WTG
- 2-exp / 2-exp
- exp-hard

gap?

2 clocks?

1WTG?
Conclusion

WTG
undec / undec
$\geq 3$ clocks / $\geq 2$ clocks

1WTG reset-acyclic
exp / exp
poly-hard

1BWTG
poly (+)
pseudo-poly (-)

1WTG?

tool?

2 clocks?

almost-divergent WTG
approx / approx
elementary complexity

divergent WTG
2-exp / 2-exp
exp-hard

gap?

Thank you!

Benjamin Monmege (Aix-Marseille Université)

Min = ○, Max = □
Conclusion

1WTG reset-acyclic
exp / exp
poly-hard

1BWTG
poly (+)
pseudo-poly (-)

1WTG?

WTG
undec / undec
\( \geq 3 \) clocks / \( \geq 2 \) clocks

almost-divergent WTG
approx / approx
elementary complexity

divergent WTG
2-exp / 2-exp
exp-hard

gap?

tool?

Thank you!

Benjamin Monmege (Aix-Marseille Université)


References II


Sketch of proof for 1BWTG

1. **Reduce the space of strategies in the 1BWTG**
   - restrict to uniform strategies w.r.t. timed behaviours

2. **Build a finite weighted game** $\mathcal{G}$
   - based on a refinement of the region abstraction

3. **Study** $\mathcal{G}$

4. **Lift results of** $\mathcal{G}$ **to the original 1BWTG**
1. Reduce the space of strategies

Intuition: no need for both players to play far from borders of regions

\[ x < 1, x := 0, 0 \]

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

Regions:
\{0\}, (0, 1), \{1\}, (1, 2), \{2\}, (2, +\infty)

Player \( \bigcirc \) wants to leave as soon as possible a state with rate \( p^+ \), and wants to stay as long as possible in a state with rate \( p^- \): so, he will always play \( \eta \)-close to a border...

Lemma:

Both players can play arbitrarily close to borders w.l.o.g.: for every \( \eta \)

\[ \text{Val}^\eta(s, v) \leq \text{Val}(s, v) \leq \text{Val}(s, v) \leq \text{Val}^\eta(s, v) \]
2. Finite weighted game abstraction

$\eta$-regions: \{0\}, (0, \eta), (1 - \eta, 1), \{1\}, (1, 1 + \eta), (2 - \eta, 2), \{2\}, (2, +\infty)$
2. Finite weighted game abstraction
3. **Study** $\mathcal{G}$: values, optimal strategies of a min-cost reachability game  

(Brihaye et al., 2016)

Optimal value: $\text{Val}_\mathcal{G}(s_1, \{0\}) = +2$ (for both players)
4. Lift results to the original 1BWTG

Reconstruct strategies in the 1BWTG from optimal strategies of $\mathcal{G}$

**Lemma:**

For all $\varepsilon > 0$, there exists $\eta > 0$ such that:

$$\text{Val}_{\mathcal{G}}(s, \{0\}) - \varepsilon \leq \text{Val}^{\eta}(s, 0) \leq \text{Val}(s, 0) \leq \overline{\text{Val}}(s, 0) \leq \overline{\text{Val}}^{\eta}(s, 0) \leq \text{Val}_{\mathcal{G}}(s, \{0\}) + \varepsilon$$
4. Lift results to the original 1BWTG

Reconstruct strategies in the 1BWTG from optimal strategies of $G$

Lemma:
For all $\varepsilon > 0$, there exists $\eta > 0$ such that:
\[
\text{Val}_G(s, \{0\}) - \varepsilon \leq \text{Val}^{\eta}(s, 0) \leq \text{Val}(s, 0) \leq \text{Val}(s, 0) \leq \text{Val}^{\eta}(s, 0) \leq \text{Val}_G(s, \{0\}) + \varepsilon
\]

- So $\text{Val}(s, 0) = \text{Val}(s, 0)$, i.e., determination
- $\varepsilon$-optimal strategies for both players
  - Finite memory for player $\bigcirc$ (finite memory in finite weighted games)
  - Infinite memory for player $\square$ (even though memoryless in finite weighted games), because it needs to ensure convergence of its differences between the 1BWTG and $G$
- Overall complexity: pseudo-polynomial (polynomial if non-negative costs) in the size of $G$, which is polynomial in the 1BWTG (because 1 clock)