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

Benjamin Monmege, Aix-Marseille Université

MoRe 2018, Oxford
Motivation: quantitative aspects of real-time synthesis

\[
\text{Environment} \parallel \text{Controller}\text{??} \models \text{Spec}
\]
Motivation: quantitative aspects of real-time synthesis

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

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

\[
\text{Environment} \parallel \text{Controller} \Rightarrow \text{Spec}
\]

Two-player game

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

Two-player \textbf{timed} game

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

\[
\begin{align*}
\text{Environment} & \parallel \text{Controller} \equiv \text{Spec} \\
\text{Two-player game}
\end{align*}
\]

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

Two-player \textbf{timed} game

Among all \textit{valid} controllers, choose a \textit{cheap/efficient} one

Two-player \textbf{weighted} timed game
Motivation: quantitative aspects of real-time synthesis

\[
\begin{align*}
\text{Environment} \parallel \text{Controller??} & \models \text{Spec} \\
\text{Two-player game}
\end{align*}
\]

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

Two-player \textit{timed} game

Among all valid controllers, choose a \textit{cheap/efficient} one

Two-player \textbf{weighted} timed game

Additional difficulty: \textbf{negative weights}

$\implies$ to model production/consumption of resources
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

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

Peak-hour ☀

- Rate: total power × 15 c€/h

Offpeak-hour ☽

- 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

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
Modelling via weighted timed games

<table>
<thead>
<tr>
<th></th>
<th>Peak-hour</th>
<th>Offpeak-hour</th>
<th>Solar panels</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rate</strong></td>
<td>15 c€/kWh</td>
<td>12 c€/kWh</td>
<td>Reselling: 20 c€/kWh</td>
</tr>
<tr>
<td><strong>States</strong></td>
<td>total power $\times$ 15 c€/h</td>
<td>total power $\times$ 12 c€/h</td>
<td>$-0.5 \times 20$ 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...)
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

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

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

Weighted graph with vertices partition between 2 players + reachability objective

Weight of a path:
\{ + \infty \text{ if not reached} \\
\text{total weight until otherwise} \}
Weighted games

Weighted graph with vertices partition between 2 players + reachability objective

\[ \min = \bigcirc, \max = \square \]
Weighted games

Weighted graph with vertices partition between 2 players + reachability objective

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

Weighted graph with vertices partition between 2 players + reachability objective

Weight of a path:
- $\{+\infty\}$ if not reached
- total weight until otherwise

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

Weighted graph with vertices partition between 2 players + reachability objective

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

Benjamin Monmege (Aix-Marseille Université) Min = $\bigcirc$, Max = $\square$
Weighted games

Weighted graph with vertices partition between 2 players + reachability objective

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

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

Weighted graph 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

$\text{Min} = \bigcirc, \text{Max} = \square$
Weighted timed games

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

(s₁, 0)
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:
\[ \text{total weight until not reached otherwise} \]

\[ (s_1, 0) \xrightarrow{0.4} (s_4, 0.4) \]

Benjamin Monmege (Aix-Marseille Université)

Min = ○, Max = □
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:
\[
\begin{array}{c}
\text{if not reached} \\
\text{total weight until otherwise}
\end{array}
\]

Benjamin Monmege (Aix-Marseille Université) Min = O, Max = □
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:
- \( \{ + \infty \text{ if not reached} \) total weight until otherwise

Benjamin Monmege (Aix-Marseille Université)

\( \text{Min} = \bigcirc, \text{Max} = \square \)
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:
\[
\begin{align*}
(s_1, 0) & \xrightarrow{0.4, \searrow} (s_4, 0.4) \xrightarrow{0.6, \rightarrow} (s_5, 0) \xrightarrow{1.5, \leftarrow} (s_4, 0) \xrightarrow{1.1, \rightarrow} (s_5, 0) \xrightarrow{2, \uparrow}(\checkmark, 2) \\
1 \times 0.4 + 1 & \quad -3 \times 0.6 + 0 \quad +1 \times 1.5 + 0 \quad -3 \times 1.1 + 0 \quad +1 \times 2 + 2 \quad = 1.8
\end{align*}
\]

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

Benjamin Monmege (Aix-Marseille Université)
Strategies and objectives

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

Min = ○, Max = □
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
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

Main object of interest:

$$\text{Val}(s, \nu) = \inf_{\sigma_{\text{Min}} \in \text{Strat}^{\text{Min}}} \sup_{\sigma_{\text{Max}} \in \text{Strat}^{\text{Max}}} \text{Weight}(\text{Exec}(s, \nu, \sigma_{\text{Min}}, \sigma_{\text{Max}})) \in \overline{\mathbb{R}}$$

What weight can players guarantee? Following which strategies?

Benjamin Monmege (Aix-Marseille Université) Min $= \bigcirc$, Max $= \square$
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)

$F_{\leq K}^\square$: $\exists$ a strategy in the weighted game for player $\bigcirc$ reaching $\checkmark$ 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 -1, 0, -W, and 0.](image)
State of the art: weighted games (shortest-path objective)

$F_{\leq K} \models : \exists$ a strategy in the weighted game for player $\bigcirc$ reaching $\checkmark$ 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?

$\bigcirc$ needs memory!
Value $-\infty$: detection is as hard as solving parity games ($\mathbf{NP} \cap \mathbf{co-NP}$)
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 $F^i(+\infty)$...

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

horizon 0: $+\infty$ $+\infty$
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-poly) memory to play optimally (but has counter strategies), 2 has optimal memoryless strategy.

Benjamin Monmege (Aix-Marseille Université) $\text{Min} = 9/33$, $\text{Max} = 29/33$
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 $F^i(+\infty)$...

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

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

- Min may require (pseudo-polynomial) memory to play optimally (but has counter strategies),
- Max has optimal memoryless strategy.

Benjamin Monmege (Aix-Marseille Université)

Min = $\#$, Max = 2

<table>
<thead>
<tr>
<th>Horizon</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$+\infty$</td>
<td>$+\infty$</td>
</tr>
<tr>
<td>1</td>
<td>$+\infty$</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$-1$</td>
<td>0</td>
</tr>
<tr>
<td>$W$</td>
<td>$-W$</td>
<td>0</td>
</tr>
</tbody>
</table>

Diagram:

- Red square node: $-1$
- Blue circle node: $0$
- Green checkmark node: $0$
- Black square node: $+\infty$
- Black circle node: $+\infty$
- Black diamond node: $0$
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 \( \mathcal{F}^i(\infty) \)

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

---

Theorem: We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players: \( \min = #, \max = 2 \frac{9}{33} \)

---

Benjamin Monmege (Aix-Marseille Université)

---

The diagram illustrates the transition probabilities and rewards for different horizons:

- Horizon 0: \( +\infty \quad +\infty \)
- Horizon 1: \( +\infty \quad 0 \)
- Horizon 2: \( -1 \quad 0 \)
- Horizon 3: \( -1 \quad -1 \)
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 $\mathcal{F}^i(+\infty)$...

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

---

strategy of

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.

Benjamin Monmege (Aix-Marseille Université) Min $= \#$, Max $= 2$
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 $F_i(+\infty)$...

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

Theorem: We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players: 2 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 $\mathcal{F}^i(\infty)$...

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

strategy of

Benjamin Monmege (Aix-Marseille Université) Min = ○, Max = □
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 $F^i(+\infty)$...

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

**Theorem:**

We can compute in pseudo-polynomial time the value of a weighted game, as well as optimal strategies for both players: $\bigcirc$ may require (pseudo-polynomial) memory to play optimally (but has counter strategies), $\square$ 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 $\geq 1$
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.
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.
Divergent weighted games analysis

\[ p \geq 1 \quad \text{and} \quad -q \leq -1 \]
Divergent weighted games analysis

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

$p \geq 1$

$-q \leq -1$

$p \geq 1$

$-q \leq -1$

$\alpha$

$\beta$

Benjamin Monmege (Aix-Marseille Université) Min $= \bigcirc$, Max $= \square$
Divergent weighted games analysis

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

divergence property

class decision

value computation

Min = ○, Max = □
Detect and remove $+\infty$ vertices (outside of the attractor of player $\bigcirc$ 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 $\bigcirc$ toward $\checkmark$)
- 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 $\Box$ toward $\checkmark \Rightarrow -\infty$
- The "value iteration" algorithm converges in linear time with initialisation at $-\infty$
Example

\[
\begin{array}{l}
\ar{v_1} \rightarrow \ar{v_2} \rightarrow [\ar{v_3} \rightarrow \ar{v_4}] \rightarrow \ar{v_5} \rightarrow \ar{v_6} \rightarrow \ar{v_7} \rightarrow \ar{v_8} \rightarrow \ar{v_f} \\
\ar{v_1} \rightarrow \ar{v_2} \rightarrow [\ar{v_3} \rightarrow \ar{v_4}] \rightarrow \ar{v_5} \rightarrow \ar{v_6} \rightarrow \ar{v_7} \rightarrow \ar{v_8} \rightarrow \ar{v_f} \\
\end{array}
\]
Example

\[ v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_f \]

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

Benjamin Monmege (Aix-Marseille Université)

Min = ⬤, Max = ☐
Example
Example

$\infty$

$\infty$

$v_1$
$v_2$
$v_3$
$v_4$
$v_5$
$v_6$
$v_7$
$v_8$
$v_9$
$v_f$

$-1$
$-1$
$-1$
$-1$
$-1$
$-10$
$1$
$0$
$0$
$1$
$1$
$2$
$2$

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

Benjamin Monmege (Aix-Marseille Université)
Example

\[\begin{align*}
\min &= \bigcirc, \quad \max &= \square
\end{align*}\]
Example

$\min = \bigcirc$, $\max = \square$
Example

Benjamin Monmege (Aix-Marseille Université)

Min = ●, Max = □
Example

Min = ○, Max = □
Part II : Weighted \textit{timed} games
State of the art

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

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

- **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))
State of the art

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

- **One-player case** (*Weighted timed automata*): optimal reachability problem is \( \text{PSPACE}-\text{complete} \)
  - 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

\( F_{\leq K} \checkmark: \exists \) 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 \text{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

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

- One-player case (\textit{Weighted timed automata}): optimal reachability problem is \textit{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: \textit{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 \textit{non-negative weights and strictly non-Zeno weight cycles}: 2-exponential algorithm (Bouyer et al., 2004b; Alur et al., 2004a)

- \textbf{One-clock} WTGs with \textit{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)

![Graphical representation of the region](image)

Fig. 6 indicates the partition induced by the relation $<\varepsilon$ for the timed automaton of Fig. 2.

Given a timed automaton $A$, its $\varepsilon$-equivalence $\equiv_\varepsilon$ is extended to the states of $S$ as done previously with $\equiv_\tau$. Let ut introduce the following notation:

- $\bar{\nu}_i < \varepsilon$ iff $\bar{\nu}′_i < \varepsilon$ for all $i \in \{1, \ldots, n\}$ with $\nu_i \leq c_i$;
- $1 - \varepsilon < \bar{\nu}_i$ iff $1 - \varepsilon < \bar{\nu}′_i$ for all $i \in \{1, \ldots, n\}$.

Remark 14. $R$ is a bounded region. We observe that the fractional parts $\bar{x}_i$ of $x_i$ are close enough to $x_i$. Similarly, if $\bar{r} = \bar{\nu}′_i - \varepsilon$ and $\bar{r}′ = \bar{\nu}′_i$, then $\bar{r} \approx \bar{r}′$.

Using the representation introduced in Remark 5, we can visualize how the partitions induced by the $\varepsilon$-equivalence are constructed. This graphical representation of the region graph of a timed automaton $A$ is very helpful in the proofs below.

Notice that the sets $Low(\nu_1)$ and $High(\nu_1)$ are disjoint since $\varepsilon \leq 1$.
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 states \( \{ p^-, p^+ \} \) included in \( \{0, +d, -d\} \)
\((d \in \mathbb{N}) (\text{no assumption on costs of transitions})\)

\[
\begin{align*}
x < 1, x & := 0, 0 \\
x > 0, x & := 0, 0 \\
x \leq 2, x & := 0, 0 \\
x \geq 1, x & := 0, 0 \\
x \leq 2, x & := 0, 0 \\
x \geq 1, x & := 0, 0
\end{align*}
\]
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 states \(\{p^-, p^+\}\) included in \(\{0, +d, -d\}\) \((d \in \mathbb{N})\) (no assumption on costs of transitions)

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

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

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

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

**Theorem:**
Computation of the value functions \( \text{Val}(s, \cdot) \) 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)
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...

\[
\begin{align*}
0 & \quad x \leq 1 \\
\rightarrow & \quad 1 \\
\rightarrow & \quad -1 \\
\rightarrow & \quad \checkmark
\end{align*}
\]

\[
\begin{align*}
x & = 1, x := 0 & \quad x & = 1 \\
x \leq 1 & \quad x \leq 1 \\
-1 & \quad -1
\end{align*}
\]
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 \( \frac{1}{2} \)...

\[
x = 1, x := 0 \\
0 \xrightarrow{x \leq 1} 1 \xrightarrow{-1} x = 1 \\
\]

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

\[
x \leq 1, y := 0 \\
0 \xrightarrow{x \leq 1, y := 0} 0 \xrightarrow{y = 0} 1 \xrightarrow{x = 1} 0 \xrightarrow{y = 1} 1 \xrightarrow{y = 1} \checkmark \\
\]

▶ How to push further the resolution of WTGs?

Benjamin Monmege (Aix-Marseille Université) Min = ●, Max = □
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 WTGs into simple 1-clock WTGs (SWTGs)
  - clock bounded by 1, no guards/invariants, no resets
- for SWTGs: compute value functions $\text{Val}(s, \cdot)$ for all states $s$. 

\[
\begin{align*}
v_1(x) & : 9 \quad \rightarrow \quad 3 \\
v_2(x) & : 9 \quad \rightarrow \quad 6 \\
v_3(x) & : 8 \quad \rightarrow \quad 5 \\
v_4(x) & : 5 \quad \rightarrow \quad 3 \\
v_5(x) & : 5 \quad \rightarrow \quad 1 
\end{align*}
\]
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)

\[
Val(s_4, x) = \sup_{0 \leq t \leq 1-x} 3t - 7 = 3(1 - x) - 7 = -3x - 4
\]
SWTGs with arbitrary weights

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

\[
\begin{align*}
\text{Val}(s_4, x) &= -3x - 4, \\
\text{Val}(s_7, x) &= -16(1 - x)
\end{align*}
\]
SWTGs with arbitrary weights

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

\[
\begin{align*}
\text{Val}(s_4, x) &= -3x - 4, \\
\text{Val}(s_7, x) &= -16(1 - x), \\
\text{Val}(s_3, x) &= \inf_{0 \leq t \leq 1-x} [4t + \min(-3(x + t) - 4, 6 - 16(1 - (x + t)))] = \\
&= \min(-3x - 4, 16x - 10)
\end{align*}
\]
Recursive elimination of states

- Player $\bigodot$ prefers to stay as long as possible in states with **minimal rate**
  $\rightarrow$ *add a final state allowing him to stay until the end, and make the state urgent*
Recursive elimination of states

- Player $\bigcirc$ prefers to stay as long as possible in states with \textbf{minimal rate}
  $\rightarrow$ \textit{add a final state allowing him to stay until the end, and make the state urgent}

- Player $\Box$ prefers to leave as soon as possible in states with \textbf{minimal rate}
  $\rightarrow$ \textit{make the state urgent}

\textbf{Theorem:} For every SWTG, all value functions are piecewise affine, with at most an exponential number of cutpoints (in number of states).

For general 1-clock WTGs?

$\rightarrow$ removing guards and invariants: previously used techniques work!

$\rightarrow$ removing resets: previously, bound the number of resets...
Recursive elimination of states

- Player ◯ prefers to stay as long as possible in states with **minimal rate**
  → *add a final state allowing him to stay until the end, and make the state urgent*

- Player □ prefers to leave as soon as possible in states with **minimal rate**
  → *make the state urgent*

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

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

\[
\begin{align*}
\text{Val}(s_3, x) &= -10 - 6 - 5.5 - 7 \quad \text{Val}(s_3, x) \downarrow \\
\text{Val}(s_4, x) &= -4 - 7 \quad \text{Val}(s_4, x) \downarrow \\
\text{Val}(s_5, x) &= -16 \quad \text{Val}(s_5, x) \downarrow \\
\text{Val}(s_2, x) &= -9.5 - 6 - 5.5 - 2 \quad \text{Val}(s_2, x) \downarrow \\
\text{Val}(s_1, x) &= -1 \quad \text{Val}(s_1, x) \downarrow \\
\text{Val}(s_7, x) &= 1 \quad \text{Val}(s_7, x) \downarrow
\end{align*}
\]

Benjamin Monmege (Aix-Marseille Université)
Bounding the number of resets needed is not possible

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

\[ x \leq 1 \]

\[ x = 1 \]

Player \# 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 \( \varepsilon \), \# needs to loop at least \( W + \lceil 1 / \log \varepsilon \rceil \) times before reaching √.

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

Player $\oplus$ 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

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 = \( \square \)
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 $\leq -1 \text{ or } \geq 1$

**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 $2$-$\text{EXP}$, and is $\text{EXP}$-hard.

Theorem:
Deciding if a weighted timed game is divergent is $\text{PSPACE}$-complete.
Weighted timed games analysis

\[ \begin{align*}
\geq 1 & \quad \text{divergence property} \\
\leq -1 & \quad \text{characterisation:}
\end{align*} \]
Weighted timed games analysis

\[
\begin{align*}
&\geq 1 \\
&\leq -1
\end{align*}
\]

**divergence property**

**characterisation:**

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

\[ \geq 1 \quad \leq -1 \]

\[ \text{divergence property} \]

\[ \text{characterisation:} \]

\[ \geq 1 \quad \leq -1 \]

Min = \( \bigcirc \), Max = \( \Box \)
Weighted timed games analysis

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

Min = ○, Max = □
Value computation in divergent weighted timed games

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

Positive SCC

Outside of the attractor of player 2 toward $-\infty$

The iterative algorithm converges on the other states in a number of steps linear with the region automaton's size, with $-\infty$ initialisation
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

Benjamin Monmege (Aix-Marseille Université)  Min = ◯, Max = □ 30/33
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 to do in case of undecidability?

- Adding cycles of weight $= 0$ to divergent WTG $\implies$ **Undecidable!**
What to do in case of undecidability?

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

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

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

Almost-divergent WTG: every SCC of the region automaton has all its cycles 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 has all its cycles

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 has all its cycles

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

Theorem:

- Approximation is decidable (in doubly exponential time complexity) for almost-divergent WTGs.
- We also provide a (semi-)symbolic algorithm that does not rely on an a-priori discretisation of the regions with a fixed granularity \(1/N\).
Extension in the negative case?

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

Almost-divergent WTG: every SCC of the region automaton has all its cycles either (\(\geq 1\) or \(= 0\)), or (\(\leq -1\) or \(= 0\))

Theorem:

- Approximation is decidable (in doubly exponential time complexity) for almost-divergent WTGs.
- We also provide a (semi-)symbolic algorithm that does not rely on an a-priori discretisation of the regions with a fixed granularity \(1/N\).
- circumvent the need for an SCC decomposition?
Conclusion

1BWTG

poly / pseudo-poly

(+), (-)

Min = ○, Max = □
Conclusion

1WTG reset-acyclic
exp / exp
poly-hard

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

undec / undec
⩾ 3 clocks /
⩾ 2 clocks

Thank you!

Benjamin Monmege (Aix-Marseille Université)
Conclusion

1WTG reset-acyclic
exp / exp
poly-hard

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

2-exp / 2-exp
exp-hard

divergent WTG

WTG

undec / undec
⩾ 3 clocks /
⩾ 2 clocks

gap?

Thank you!

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

1WTG reset-acyclic
exp / exp
poly-hard

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

almost-divergent WTG
approx / approx
2-exp. + symbolic algorithm

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

Thank you!

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

WTG

undec / undec
≥ 3 clocks / ≥ 2 clocks

1WTG reset-acyclic
exp / exp
poly-hard

almost-divergent WTG
approx / approx
2-exp. + symbolic algorithm

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

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

Benjamin Monmege (Aix-Marseille Université)

Min = ○, Max = □
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
- $2$-exp. + symbolic algorithm

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

1WTG?

Benjamin Monmege (Aix-Marseille Université)

Min = $\bigcirc$, Max = $\blacksquare$
Conclusion

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

almost-divergent WTG
approx / approx
2-exp. + *symbolic* algorithm

1WTG reset-acyclic
exp / exp
poly-hard

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

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

gap?

1WTG?

---

Benjamin Monmege (Aix-Marseille Université)
Conclusion

1WTG reset-acyclic
exp / exp
poly-hard

1WTG?

2 clocks?

WTG
undec / undec
≥ 3 clocks / ≥ 2 clocks

almost-divergent WTG
approx / approx
2-exp. + symbolic algorithm

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

1BWTG?

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

gap?

Benjamin Monmege (Aix-Marseille Université)
Conclusion

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

almost-divergent WTG
approx / approx
2-exp. + symbolic algorithm

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

gap?

1WTG reset-acyclic
exp / exp
poly-hard

1BWTG
poly (†) / pseudo-poly (‡)

1WTG?

2 clocks?

tool?

2 clocks?

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

WTG
undec / undec
\[ \geq 3 \text{ clocks} / \geq 2 \text{ clocks} \]

1WTG reset-acyclic
exp / exp
poly-hard

2 clocks?

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

almost-divergent WTG
approx / approx
2-exp. + symbolic algorithm

divergent WTG
2-exp / 2-exp
exp-hard
gap?

1WTG?

tool?

Thank you!

Benjamin Monmege (Aix-Marseille Université)
References I


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 $G$**
   - based on a refinement of the region abstraction

3. **Study $G$**

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

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

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

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 \( \eta \)

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

\[ \eta \text{-regions: } \emptyset, (0, \eta), (1 - \eta, 1), \{1\}, (1, 1 + \eta), (2 - \eta, 2), \{2\}, (2, +\infty) \]
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 $G$

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

$\text{Val}_G(s, \{0\}) - \varepsilon \leq \text{Val}^\eta(s, 0) \leq \text{Val}(s, 0) \leq \overline{\text{Val}}(s, 0) \leq \overline{\text{Val}}^\eta(s, 0) \leq \text{Val}_G(s, \{0\}) + \varepsilon$
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:

$$
Val_G(s, \{0\}) - \varepsilon \leq Val^\eta(s, 0) \leq Val(s, 0) \leq Val(s, 0) \leq Val^\eta(s, 0) \leq Val_G(s, \{0\}) + \varepsilon
$$

- So $Val(s, 0) = Val(s, 0)$, 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)