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

$$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$$

Motivation: quantitative aspects of real-time synthesis

$$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$$

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

Motivation: quantitative aspects of real-time synthesis

$$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$$

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

Among all *valid* controllers, choose a *cheap/efficient* one

Motivation: quantitative aspects of real-time synthesis

$$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$$

Two-player game

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

Among all *valid* controllers, choose a *cheap/efficient* one

Motivation: quantitative aspects of real-time synthesis

$$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$$

Two-player game

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

Two-player **timed** game

Among all *valid* controllers, choose a *cheap/efficient* one

Motivation: quantitative aspects of real-time synthesis

$$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$$

Two-player game

Real-time requirements/environment $\Longrightarrow$ 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

$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$

Two-player game

Real-time requirements/environment $\Longrightarrow$ 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**

$\Longrightarrow$ to model production/consumption of resources

Modelling via weighted timed games

	Peak-hour 	Offpeak-hour 
	15 c€/kWh	12 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h

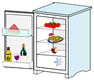
states to record which device is on/off: computation of the total power

Modelling via weighted timed games

	Peak-hour 	Offpeak-hour 
	15 c€/kWh	12 c€/kWh
rate:	total power × 15 c€/h	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 	Offpeak-hour 	Solar panels 
	15 c€/kWh	12 c€/kWh	Reselling: 20 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h	-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 	Offpeak-hour 	Solar panels 
	15 c€/kWh	12 c€/kWh	Reselling: 20 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h	-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 	Offpeak-hour 	Solar panels 
	15 c€/kWh	12 c€/kWh	Reselling: 20 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h	-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 	Offpeak-hour 	Solar panels 
	15 c€/kWh	12 c€/kWh	Reselling: 20 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h	-0.5×20 c€/h

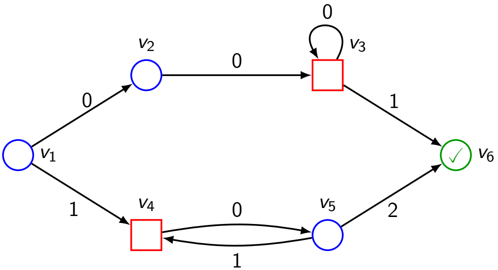
states to record which device is on/off: computation of the total power

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

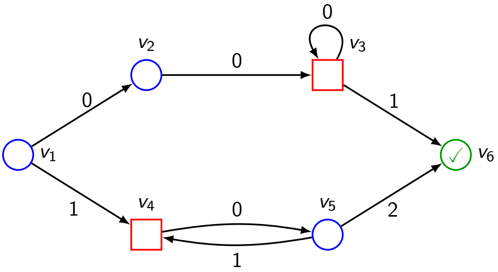
Solution 1 : discretisation of time, resolution via a *weighted game*
Solution 2 : thin time behaviours, resolution via a *weighted timed game*

Weighted games



Weighted graph with vertices partition between 2 players + reachability objective

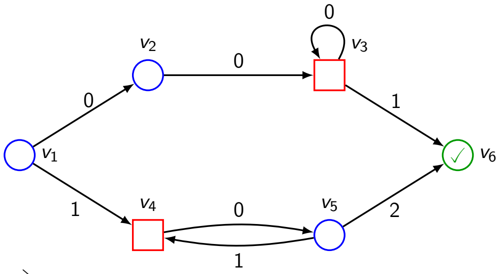
Weighted games



Weighted graph with vertices partition between 2 players
+ reachability objective

v_1

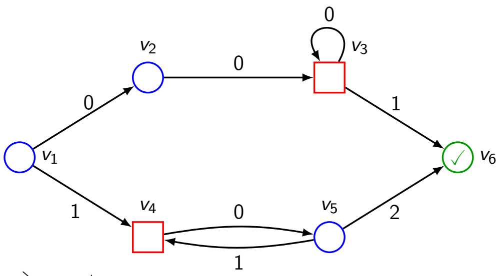
Weighted games



$v_1 \xrightarrow{\searrow} v_4$

Weighted graph with vertices partition between 2 players
+ reachability objective

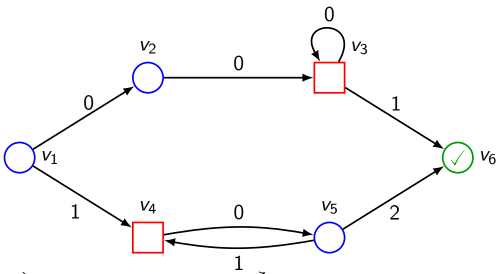
Weighted games



$v_1 \xrightarrow{\searrow} v_4 \xrightarrow{\rightarrow} v_5$

Weighted graph with vertices partition between 2 players
+ reachability objective

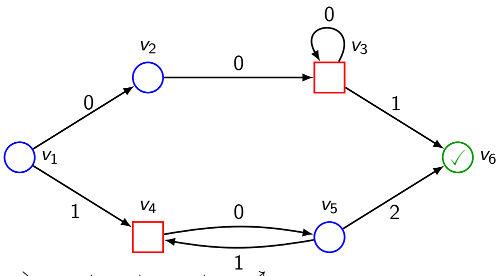
Weighted games



$v_1 \xrightarrow{\searrow} v_4 \xrightarrow{\rightarrow} v_5 \xrightarrow{\leftarrow} v_4 \xrightarrow{\rightarrow} v_5 \xrightarrow{\nearrow} \checkmark$

Weighted graph with vertices partition between 2 players
+ reachability objective

Weighted games

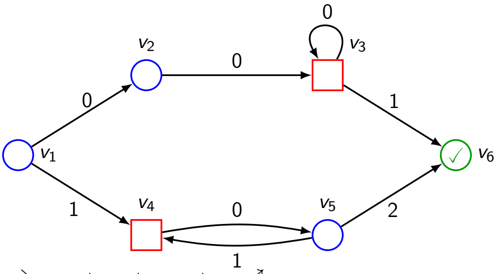


$v_1 \xrightarrow{\searrow} v_4 \xrightarrow{\rightarrow} v_5 \xrightarrow{\leftarrow} v_4 \xrightarrow{\rightarrow} v_5 \xrightarrow{\nearrow} \checkmark$

$1 \quad +1 \quad +2 \quad = 4$

Weighted graph with vertices partition between 2 players
+ reachability objective

Weighted games



Weighted graph with vertices partition between 2 players
+ reachability objective

$$v_1 \xrightarrow{\searrow} v_4 \xrightarrow{\rightarrow} v_5 \xrightarrow{\leftarrow} v_4 \xrightarrow{\rightarrow} v_5 \xrightarrow{\nearrow} \checkmark$$

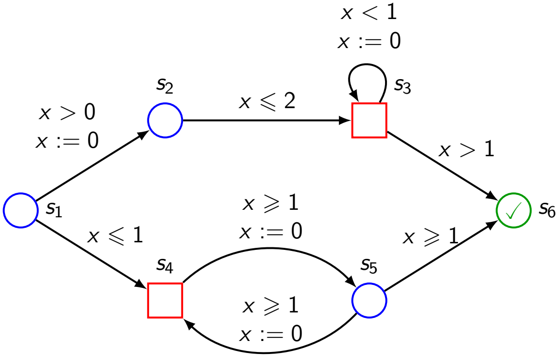
$$1 \quad +1 \quad +2 \quad = 4$$

$$v_1 \xrightarrow{\nearrow} v_2 \xrightarrow{\rightarrow} v_3 \xrightarrow{\circlearrowleft} v_3 \xrightarrow{\circlearrowleft} v_3 \quad \cdots$$

$$\cdots \quad = +\infty (\checkmark \text{ not reached})$$

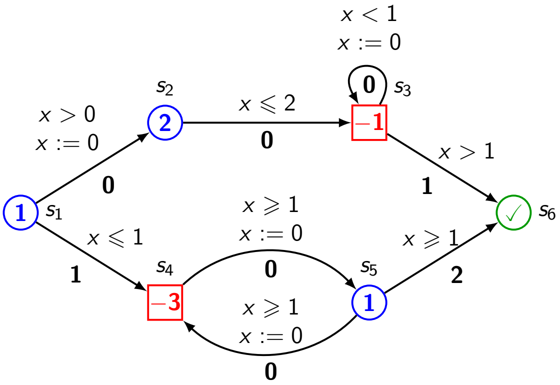
$$\text{Weight of a path: } \begin{cases} +\infty & \text{if } \checkmark \text{ not reached} \\ \text{total weight until } \checkmark & \text{otherwise} \end{cases}$$

Weighted timed games



Timed automaton
with state partition between
2 players
+ reachability objective

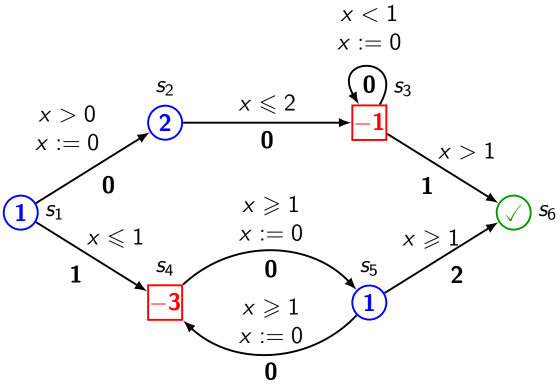
Weighted timed games



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

$(s_1, 0)$

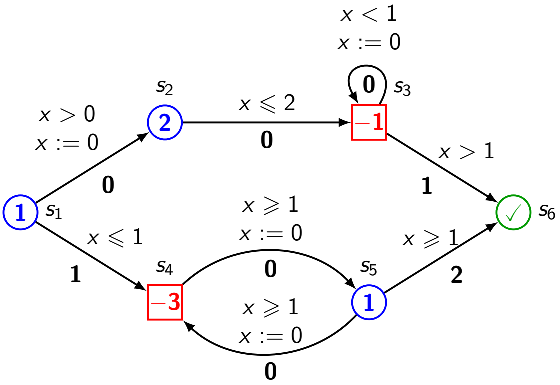
Weighted timed games



$(s_1, 0) \xrightarrow{0.4, \searrow} (s_4, 0.4)$

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

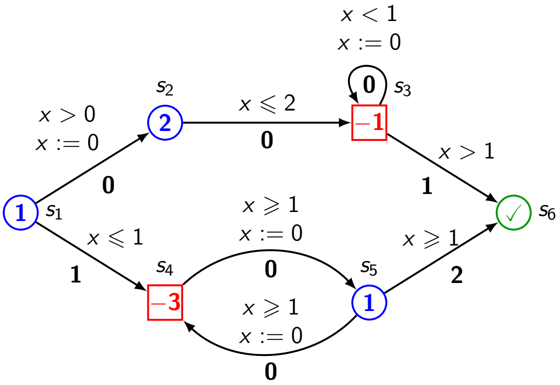
Weighted timed games



$(s_1, 0) \xrightarrow{0.4, \searrow} (s_4, 0.4) \xrightarrow{0.6, \rightarrow} (s_5, 0)$

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

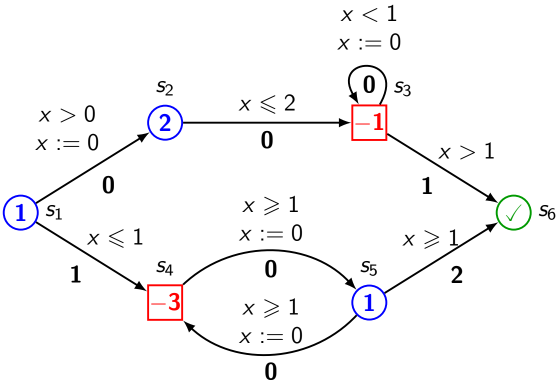
Weighted timed games



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

$$(s_1, 0) \xrightarrow{0.4, \searrow} (s_4, 0.4) \xrightarrow{0.6, \rightarrow} (s_5, 0) \xrightarrow{1.5, \leftarrow} (s_4, 0) \xrightarrow{1.1, \rightarrow} (s_5, 0) \xrightarrow{2, \nearrow} (\checkmark, 2)$$

Weighted timed games

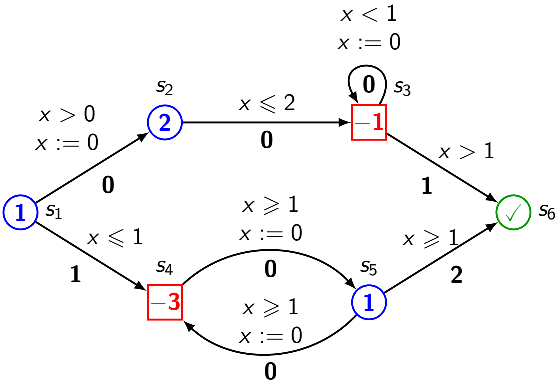


Timed automaton
with state partition between
2 players
+ reachability objective
+ linear rates on states
+ discrete weights on
transitions

$$(s_1,0)\xrightarrow{0.4,\searrow}(s_4,0.4)\xrightarrow{0.6,\rightarrow}(s_5,0)\xrightarrow{1.5,\leftarrow}(s_4,0)\xrightarrow{1.1,\rightarrow}(s_5,0)\xrightarrow{2,\nearrow}(\checkmark,2)$$

$$\mathbf{1}\times 0.4+\mathbf{1} \quad -\mathbf{3}\times 0.6+\mathbf{0} \quad +\mathbf{1}\times 1.5+\mathbf{0} \quad -\mathbf{3}\times 1.1+\mathbf{0} \quad +\mathbf{1}\times 2+\mathbf{2} \quad = 1.8$$

Weighted timed games



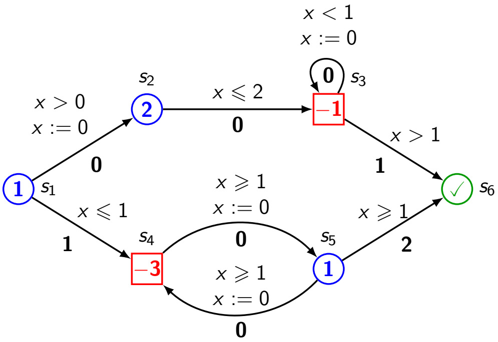
Timed automaton
with state partition between
2 players
+ reachability objective
+ linear rates on states
+ discrete weights on
transitions

$$(s_1,0)\xrightarrow[\mathbf{1}\times 0.4+\mathbf{1}]{0.4,\searrow}(s_4,0.4)\xrightarrow[-\mathbf{3}\times 0.6+\mathbf{0}]{0.6,\rightarrow}(s_5,0)\xrightarrow[+\mathbf{1}\times 1.5+\mathbf{0}]{1.5,\leftarrow}(s_4,0)\xrightarrow[-\mathbf{3}\times 1.1+\mathbf{0}]{1.1,\rightarrow}(s_5,0)\xrightarrow[+\mathbf{1}\times 2+\mathbf{2}]{2,\nearrow}(\checkmark,2) \quad = 1.8$$

$$(s_1,0)\xrightarrow[\mathbf{1}\times 0.2+\mathbf{0}]{0.2,\nearrow}(s_2,0)\xrightarrow[+\mathbf{2}\times 0.9+\mathbf{0}]{0.9,\rightarrow}(s_3,0.9)\xrightarrow[-\mathbf{1}\times 0.2+\mathbf{0}]{0.2,\circlearrowleft}(s_3,0)\xrightarrow[-\mathbf{1}\times 0.9+\mathbf{0}]{0.9,\circlearrowleft}(s_3,0)\quad\cdots\quad\cdots \quad = +\infty$$

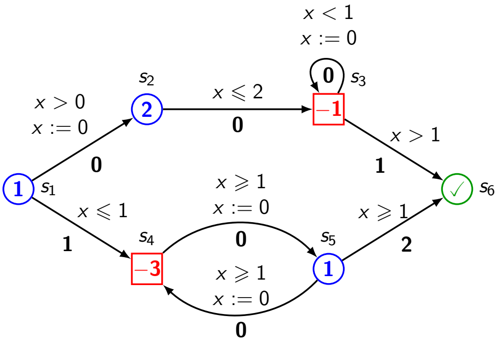
$$\text{Weight of an execution : }\begin{cases} +\infty & \text{if } \checkmark \text{ not reached}\\ \text{total weight until } \checkmark & \text{otherwise}\end{cases}$$

Strategies and objectives



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

Strategies and objectives

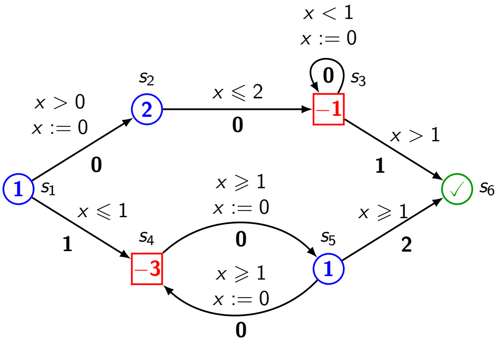


Strategy for a player: map finite executions to a delay and a transition

Objective of player ○: 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 ○: reach ✓ **and** minimise the weight
Objective of player □: avoid ✓ **or, if not possible,** maximise the weight

Main object of interest:

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

What weight can players guarantee? Following which strategies?

Part I : Weighted games

State of the art: weighted games (shortest-path objective)

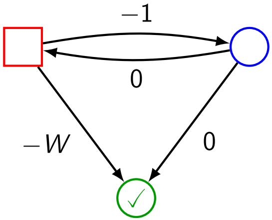
$\mathsf{F}_{\leqslant K}\checkmark$: $\exists$ a strategy in the weighted game for player $\bigcirc$ reaching $\checkmark$ with a cost $\leqslant K$?

- 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)

$\mathsf{F}_{\leqslant K}\checkmark$: $\exists$ a strategy in the weighted game for player $\bigcirc$ reaching $\checkmark$ with a cost $\leqslant K$?

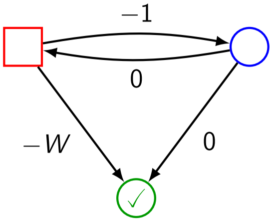
- one-player: shortest path in a weighted graph... polynomial algo.
- two players, non-negative weights only: polynomial algo.
  "Dijkstra algorithm on 2 players games" (Khachiyan et al., 2008)
- two players, arbitrary weights?



State of the art: weighted games (shortest-path objective)

$\mathsf{F}_{\leqslant K}\checkmark$: $\exists$ a strategy in the weighted game for player $\bigcirc$ reaching $\checkmark$ with a cost $\leqslant K$?

- 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?



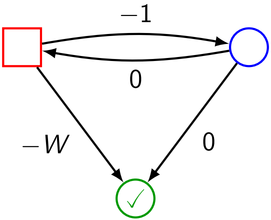
$\bigcirc$ needs memory!
Value $-\infty$: detection is as hard as solving parity games (**NP** $\cap$ **co-NP**)

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 $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(\boldsymbol{x})_v = \begin{cases} \min\limits_{e=(v,a,v')\in E} \big(\mathsf{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathsf{Min}} \\ \max\limits_{e=(v,a,v')\in E} \big(\mathsf{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathsf{Max}} \end{cases}$$



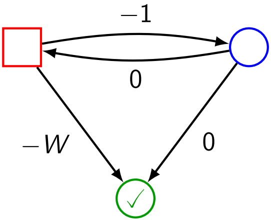
	□	○
horizon 0:	$+\infty$	$+\infty$

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 $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(\boldsymbol{x})_v = \begin{cases} \min\limits_{e=(v,a,v')\in E} \big(\mathsf{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathsf{Min}} \\ \max\limits_{e=(v,a,v')\in E} \big(\mathsf{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathsf{Max}} \end{cases}$$



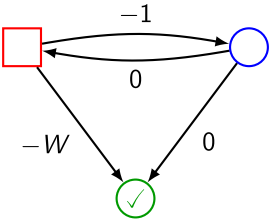
	□	○
horizon 0:	$+\infty$	$+\infty$
horizon 1:	$+\infty$	0

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 $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(\boldsymbol{x})_v = \begin{cases} \min\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Min}} \\ \max\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Max}} \end{cases}$$



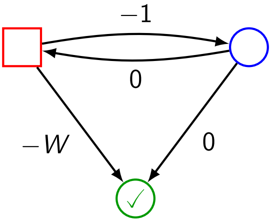
	$\square$	$\bigcirc$
horizon 0:	$+\infty$	$+\infty$
horizon 1:	$+\infty$	0
horizon 2:	-1	0

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 $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(\boldsymbol{x})_v = \begin{cases} \min\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Min}} \\ \max\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Max}} \end{cases}$$



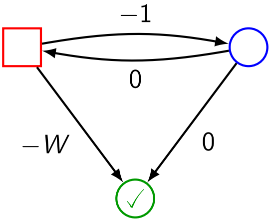
	□	○
horizon 0:	$+\infty$	$+\infty$
horizon 1:	$+\infty$	0
horizon 2:	-1	0
horizon 3:	-1	-1

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 $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(\boldsymbol{x})_v = \begin{cases} \min\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Min}} \\ \max\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Max}} \end{cases}$$



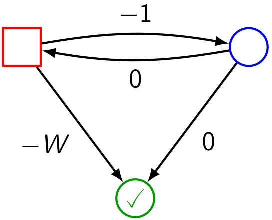
	□	○
horizon 0:	$+\infty$	$+\infty$
horizon 1:	$+\infty$	0
horizon 2:	-1	0
horizon 3:	-1	-1
horizon 4:	-2	-1

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 $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(\boldsymbol{x})_v = \begin{cases} \min\limits_{e=(v,a,v')\in E} \big(\mathsf{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathsf{Min}} \\ \max\limits_{e=(v,a,v')\in E} \big(\mathsf{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathsf{Max}} \end{cases}$$



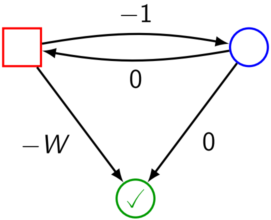
	□	○
horizon 0:	$+\infty$	$+\infty$
horizon 1:	$+\infty$	0
horizon 2:	-1	0
horizon 3:	-1	-1
horizon 4:	-2	-1
	...	...
horizon $2W+1$:	$-W$	$-W$

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 $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(\boldsymbol{x})_v = \begin{cases} \min\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Min}} \\ \max\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Max}} \end{cases}$$



	□	○
horizon 0:	$+\infty$	$+\infty$
horizon 1:	$+\infty$	0
horizon 2:	-1	0
horizon 3:	-1	-1
horizon 4:	-2	-1
	...	...
horizon $2W+1$:	$-W$	$-W$
horizon $2W+2$:	$-W$	$-W$

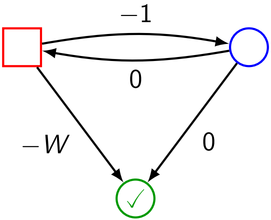
strategy of ○

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 $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(\boldsymbol{x})_v = \begin{cases} \min\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Min}} \\ \max\limits_{e=(v,a,v')\in E} \big(\mathrm{Weight}(e) + \boldsymbol{x}_{v'}\big) & \text{if } v \in V_{\mathrm{Max}} \end{cases}$$



	□	○
horizon 0:	$+\infty$	$+\infty$
horizon 1:	$+\infty$	0
horizon 2:	-1	0
horizon 3:	-1	-1
horizon 4:	-2	-1
	...	...
horizon $2W+1$:	$-W$	$-W$
horizon $2W+2$:	$-W$	$-W$

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), □ 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 $\leqslant -1$ or $\geqslant 1$

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 $\leqslant -1$ or $\geqslant 1$

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 $\leqslant -1$ or $\geqslant 1$

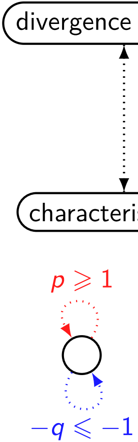
Theorem:

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.

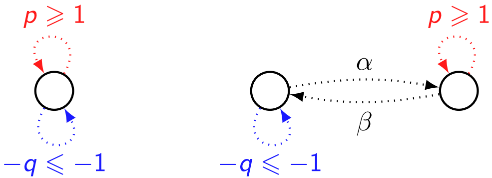
Divergent weighted games analysis



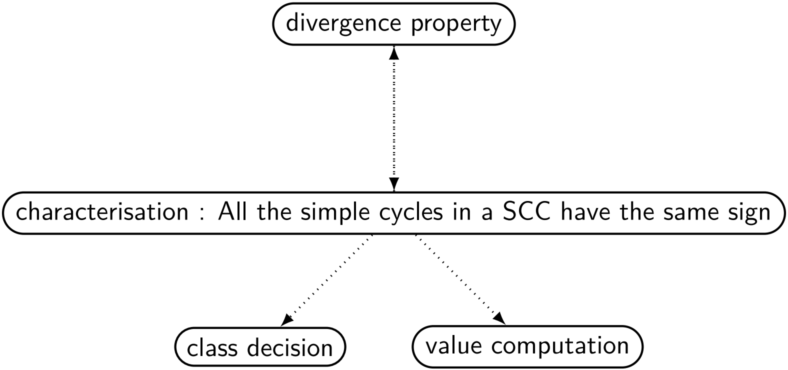
Divergent weighted games analysis

divergence property

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



Divergent weighted games analysis



Value computation in a divergent weighted game

- 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

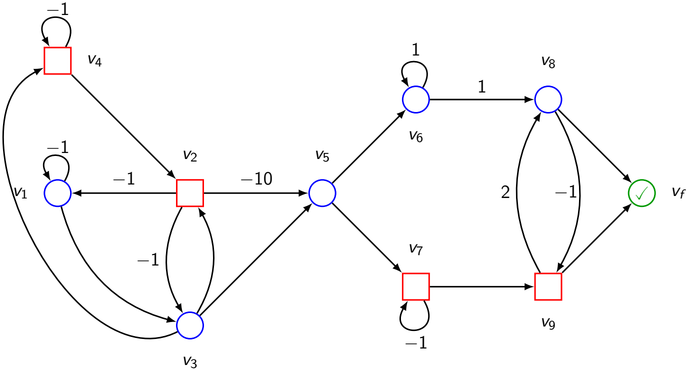
positive SCC

- The "value iteration" algorithm converges in linear time

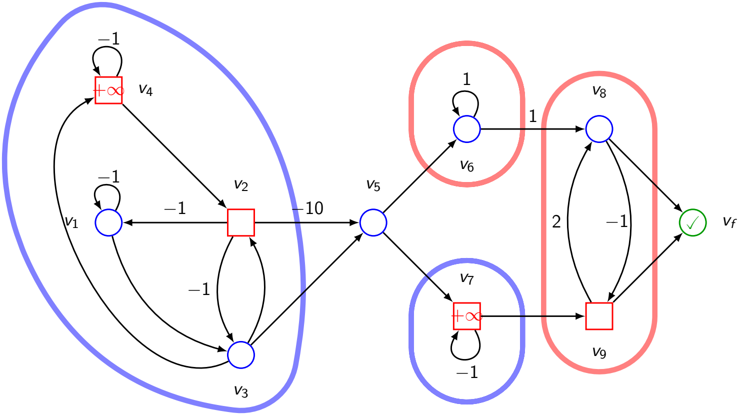
negative SCC

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

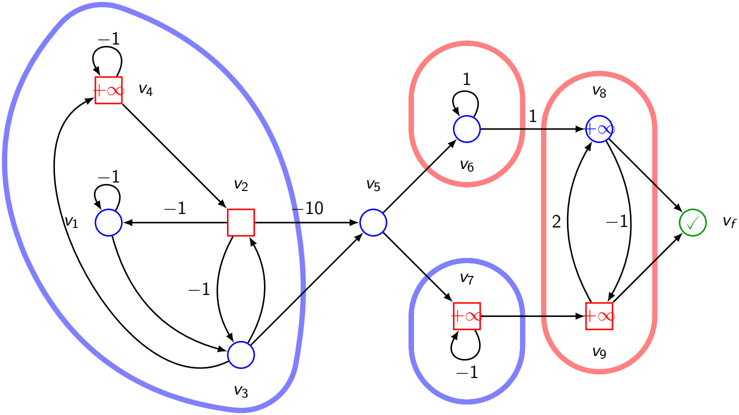
Example



Example

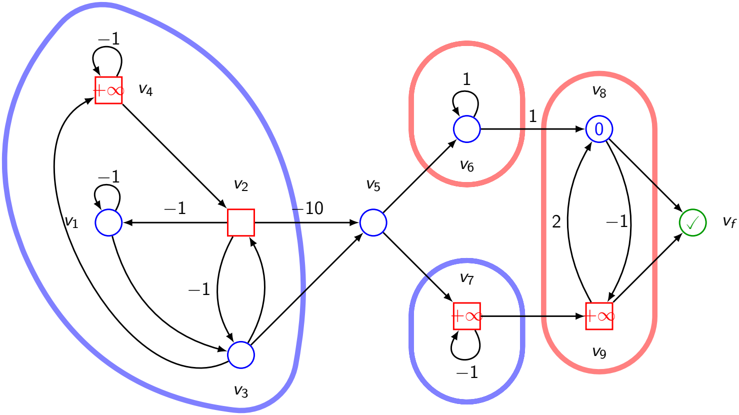


Example

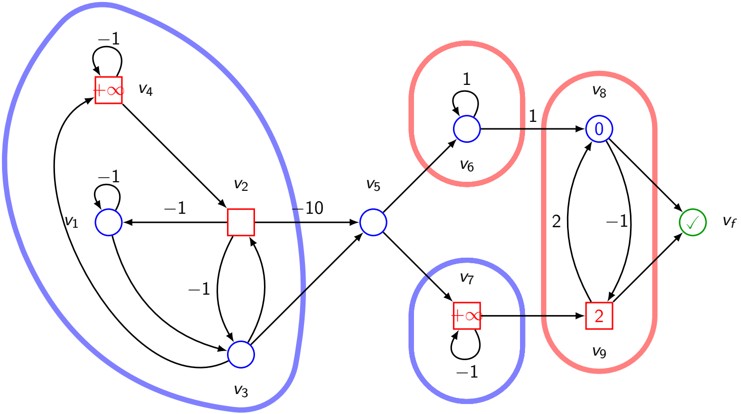


-1
$+\infty$
v_4
-1
v_1
-1
v_2
-10
v_5
-1
v_3
1
1
v_6
v_7
$+\infty$
-1
v_8
$+\infty$
2
-1
$+\infty$
v_9
$\checkmark$
v_f

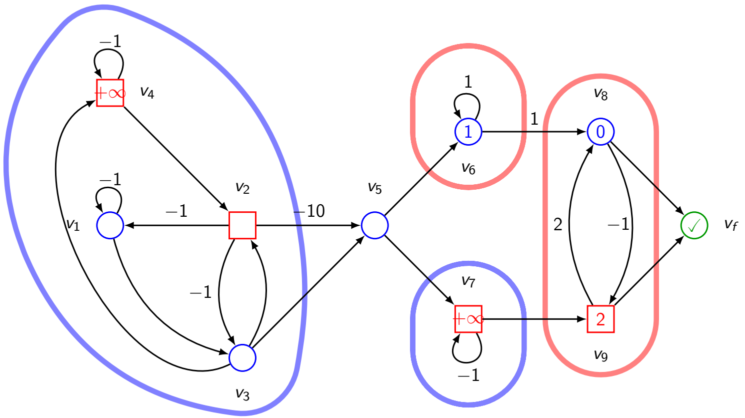
Example



Example

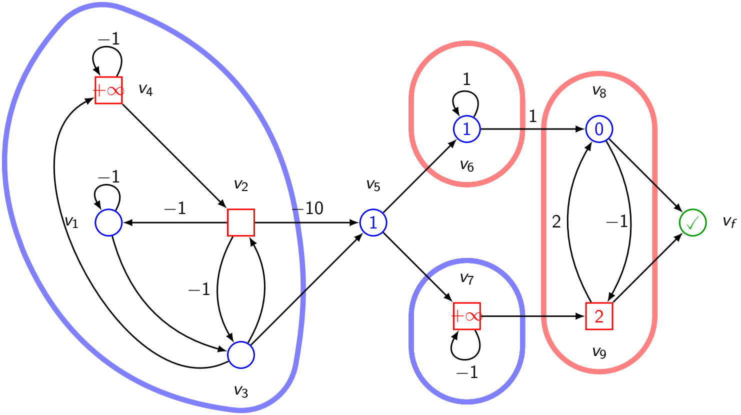


Example



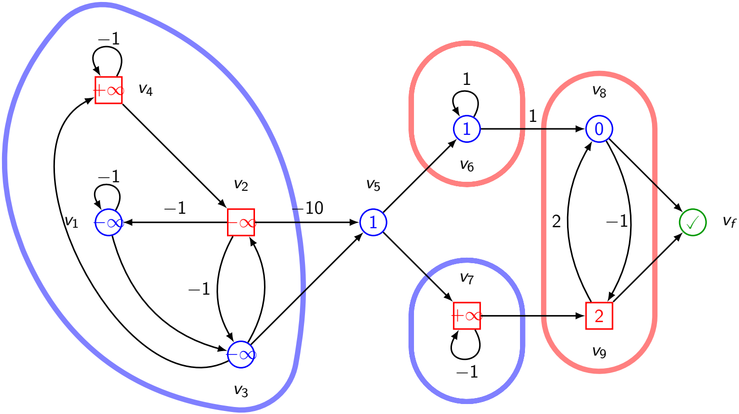
-1
$+\infty$
v_4
-1
v_1
-1
v_2
-10
v_5
-1
v_3
1
1
v_6
v_8
0
v_7
$+\infty$
-1
2
-1
2
v_9
$\checkmark$
v_f

Example



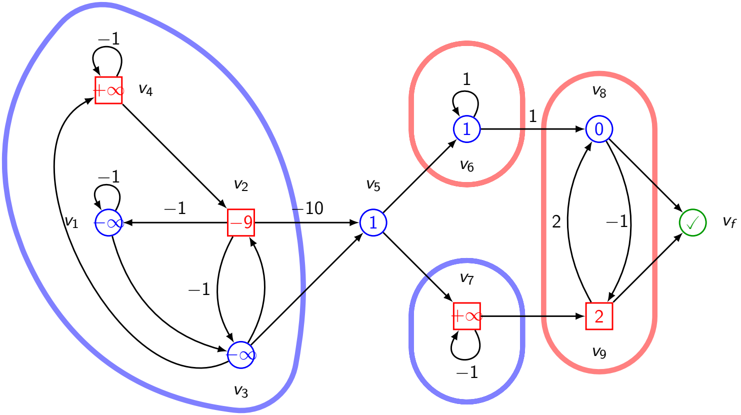
-1
$+\infty$
v_4
-1
v_1
-1
v_2
-10
-1
v_3
v_5
1
1
1
v_6
v_7
$+\infty$
-1
v_8
0
2
-1
2
v_9
$\checkmark$
v_f

Example



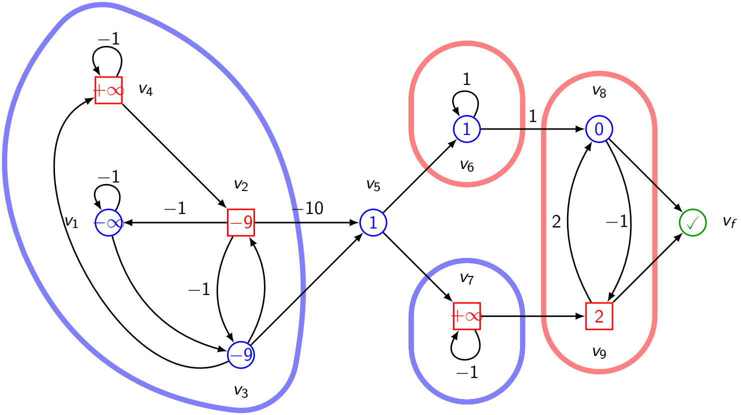
-1
$+\infty$
v_4
-1
v_1
$-\infty$
-1
v_2
$-\infty$
-10
v_5
1
-1
$-\infty$
v_3
1
1
v_6
1
v_7
$+\infty$
-1
v_8
0
2
-1
2
v_9
v_f

Example



-1
$+\infty$
v_4
-1
v_1
$-\infty$
-1
v_2
-9
-10
-1
$-\infty$
v_3
v_5
1
1
1
v_6
1
v_7
$+\infty$
-1
v_8
0
2
-1
2
v_9
$\checkmark$
v_f

Example



Part II : Weighted **timed** games

State of the art

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

State of the art

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

- 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

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

- 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

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

- 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

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

- 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

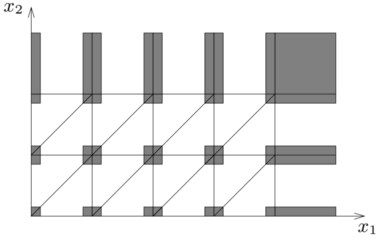
$\mathsf{F}_{\leqslant K}\checkmark$: $\exists$ a strategy in the WTG (weighted timed game) for player $\bigcirc$ reaching $\checkmark$ with a cost $\leqslant K$?

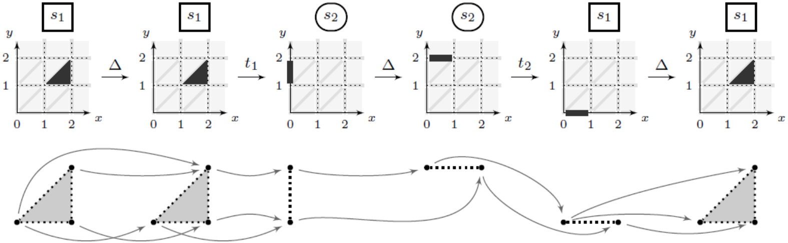
- One-player case (**Weighted timed automata**): optimal reachability problem is PSPACE-complete
  - Algorithm based on regions (Bouyer et al., 2004a, 2007);
  - and hardness shown for timed automata with at least 2 clocks (Fearnley and Jurdziński, 2013; Haase et al., 2012)
- 2-player WTGs: undecidable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



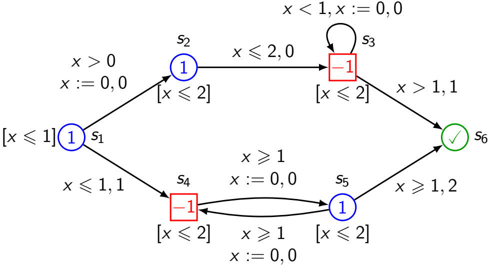


One-clock Bi-Valued WTGs (1BWTGs)

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)

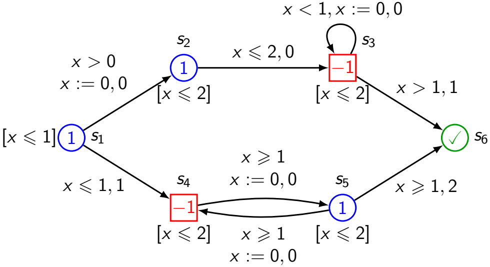
Assumption: rates of states $\{p^-, p^+\}$ included in $\{0, +d, -d\}$ ($d \in \mathbb{N}$) (no assumption on costs of transitions)



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)



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

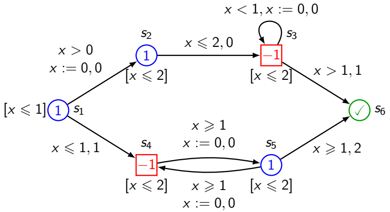
regions refined with corner information:

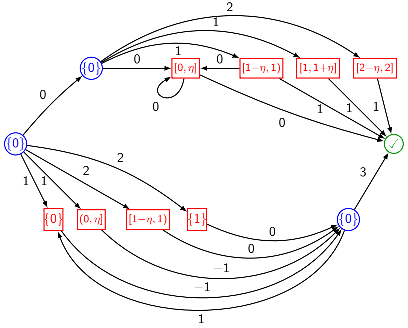
$$\{0\}, (0,\eta), (1-\eta, 1), \{1\}, (1, 1+\eta), (2-\eta, 2), \{2\}, (2, +\infty)$$

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)

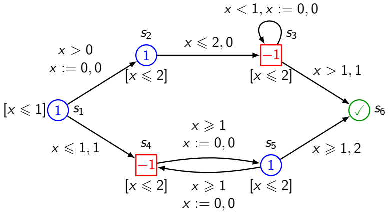


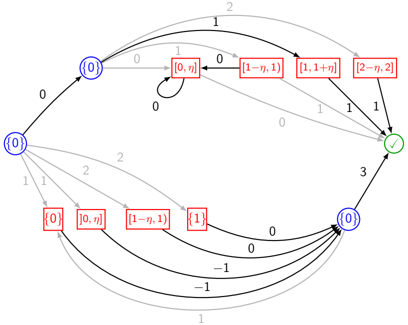


One-clock Bi-Valued WTGs (1BWTGs)

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)

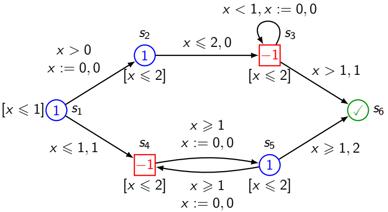


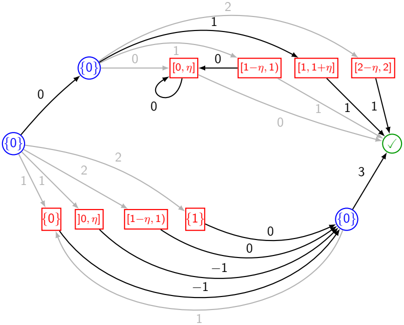


One-clock Bi-Valued WTGs (1BWTGs)

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)





Theorem:

Computation of the value functions $\mathrm{Val}(s, \cdot)$ of states of a 1BWTG and synthesis of ε-optimal strategies for ○ in pseudo-polynomial time

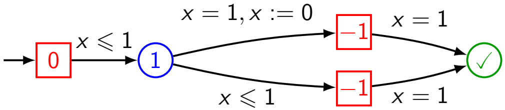
- Only non-negative costs $\Longrightarrow$ polynomial time

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 $1/2$...

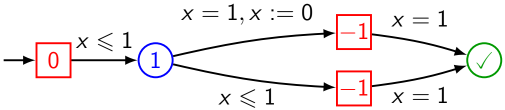


1BWTG: maximal fragment for corner-point abstraction

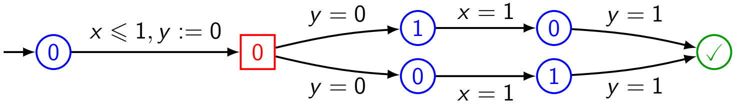
Generalisation by Engel Lefaucheux: two rates $\{p^-, p^+\}$ included in $\{0, +d, -c\}$ ($d, c \in \mathbb{N}$)

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

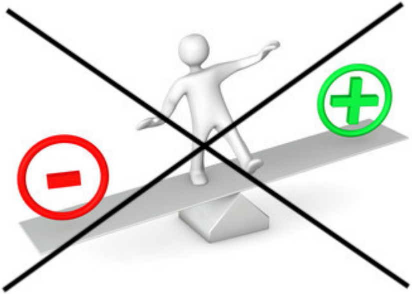
- With 3 weights in $\{-1, 0, +1\}$: value 1/2...



- 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

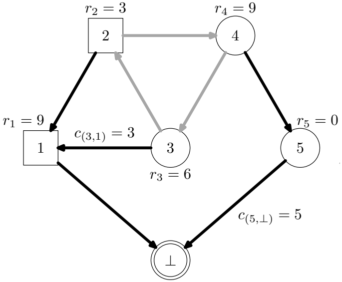
- precomputation: polynomial-time cascade of simplification of 1-clock WTGs into simple 1-clock WTGs (SWTGs)
  - clock bounded by 1, no guards/invariants, no resets

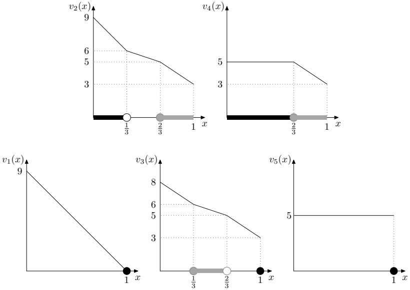
Related work: 1-clock, non-negative weights

(Hansen et al., 2013): strategy improvement algorithm

(Bouyer et al., 2006b; Rutkowski, 2011): iterative elimination of locations

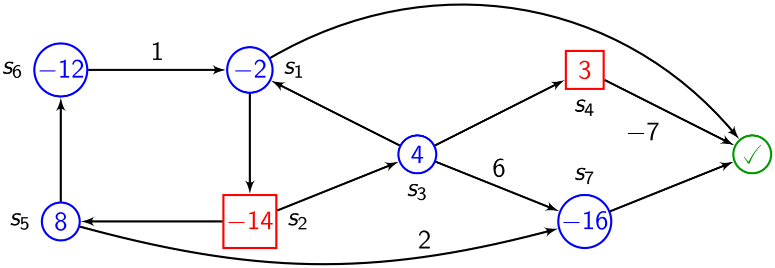
- precomputation: polynomial-time cascade of simplification of 1-clock WTGs into simple 1-clock WTGs (SWTGs)
  - clock bounded by 1, no guards/invariants, no resets
- for SWTGs: compute value functions $\mathsf{Val}(s,\cdot)$ for all states s.





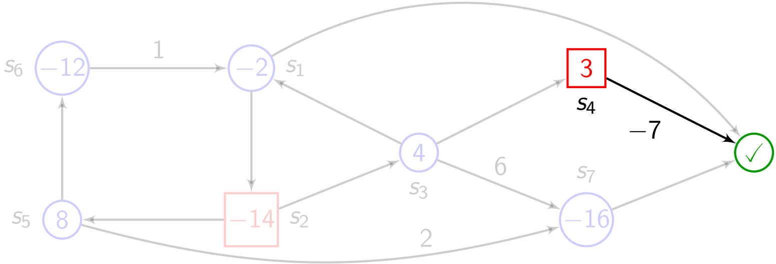
SWTGs with arbitrary weights

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



SWTGs with arbitrary weights

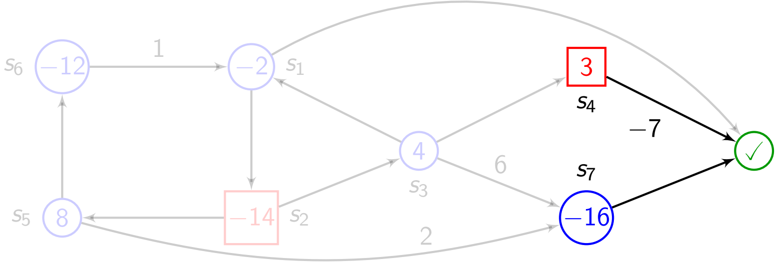
Joint work with Thomas Brihaye, Gilles Geeraerts, Axel Haddad and Engel Lefaucheux (Brihaye et al., 2015)



$$\mathrm{Val}(s_4, x) = \sup_{0 \leqslant t \leqslant 1-x} 3t - 7 = 3(1-x) - 7 = -3x - 4$$

SWTGs with arbitrary weights

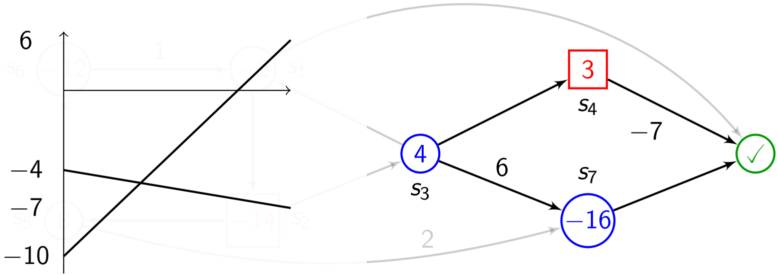
Joint work with Thomas Brihaye, Gilles Geeraerts, Axel Haddad and Engel Lefaucheux (Brihaye et al., 2015)



$$\mathsf{Val}(s_4, x) = -3x - 4, \qquad \mathsf{Val}(s_7, x) = -16(1 - x)$$

SWTGs with arbitrary weights

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



$$\mathrm{Val}(s_4, x) = -3x - 4, \qquad \mathrm{Val}(s_7, x) = -16(1-x),$$
$$\mathrm{Val}(s_3, x) = \inf_{0 \leqslant t \leqslant 1-x}[4t + \min(-3(x+t) - 4, 6 - 16(1-(x+t)))] = \min(-3x-4, 16x-10)$$

Recursive elimination of states

- Player ○ 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 **minimal rate**
  $\rightarrow$ *add a final state allowing him to stay until the end, and make the state urgent*
- Player $\square$ prefers to leave as soon as possible in states with **minimal rate**
  $\rightarrow$ *make the state urgent*

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*

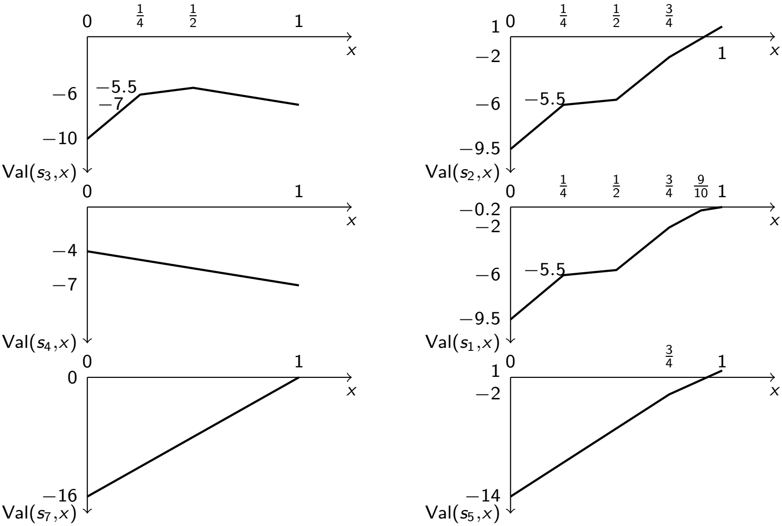
Theorem:

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

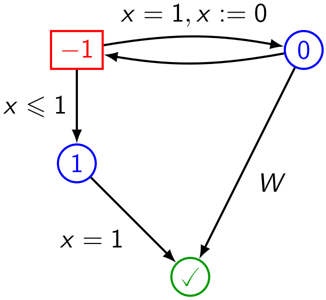
For general 1-clock WTGs?

- removing guards and invariants: previously used techniques work!
- removing resets: previously, bound the number of resets...

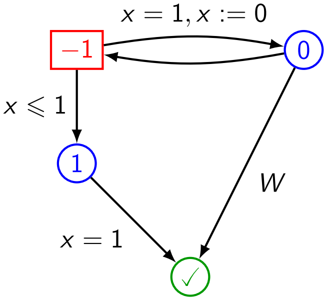
Solving SWTGs with arbitrary weights



Bounding the number of resets needed is not possible

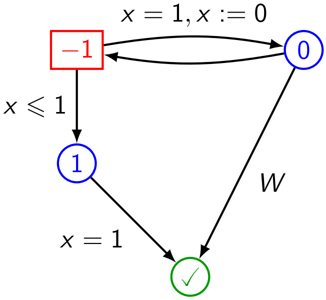


Bounding the number of resets needed is not possible



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

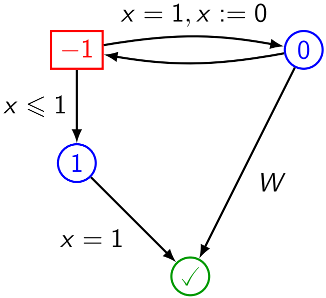
Bounding the number of resets needed is not possible



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

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

Bounding the number of resets needed is not possible

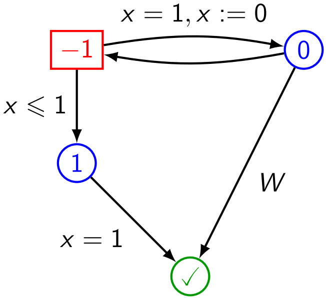


Player ○ can guarantee (i.e., ensure to be below) value ε for all $\varepsilon > 0$...

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

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

Bounding the number of resets needed is not possible



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

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

... moreover, to obtain ε, $\bigcirc$ 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

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 $\mathcal{F}^i(+\infty)$...

$$\mathcal{F}(\boldsymbol{x})_{(s,\nu)} = \begin{cases} \sup\limits_{(s,\nu)\xrightarrow{d,t}(s',\nu')} \big(d \times \mathrm{Weight}(s) + \mathrm{Weight}(t) + \boldsymbol{x}_{(s',\nu')}\big) & \text{if } s \in S_{\mathrm{Max}} \\ \inf\limits_{(s,\nu)\xrightarrow{d,t}(s',\nu')} \big(d \times \mathrm{Weight}(s) + \mathrm{Weight}(t) + \boldsymbol{x}_{(s',\nu')}\big) & \text{if } s \in S_{\mathrm{Min}} \end{cases}$$

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 $\leqslant -1$ or $\geqslant 1$

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 $\leqslant -1$ or $\geqslant 1$

Theorem:

The value problem on divergent weighted timed games is in 2-**EXP**, and is **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 $\leqslant -1$ or $\geqslant 1$

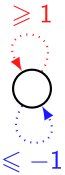
Theorem:

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

Theorem:

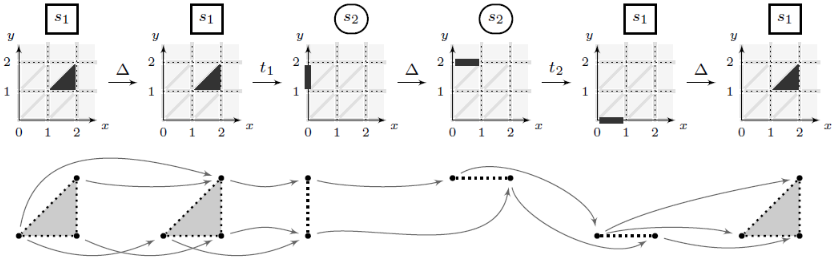
Deciding if a weighted timed game is divergent is **PSPACE**-complete.

Weighted timed games analysis

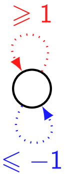


divergence property

characterisation :



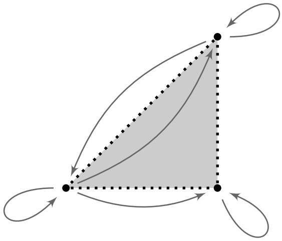
Weighted timed games analysis



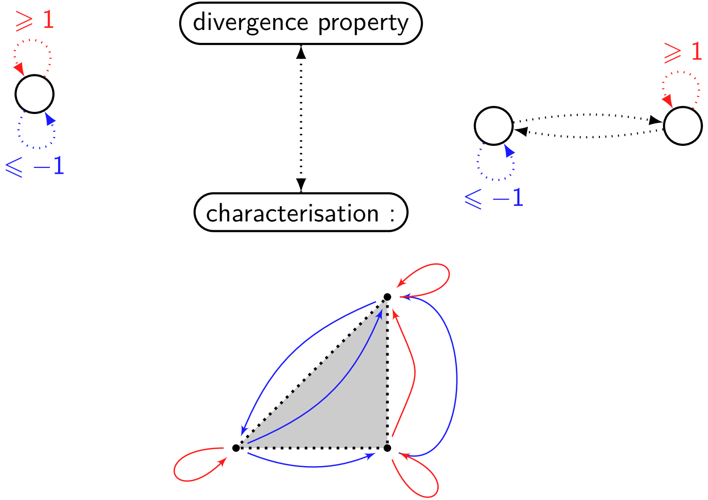
divergence property

$\updownarrow$

characterisation :



Weighted timed games analysis

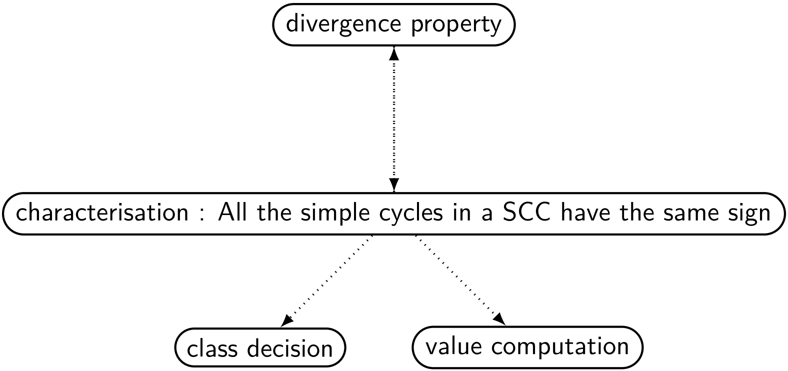


Weighted timed games analysis

divergence property

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

Weighted timed games analysis



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

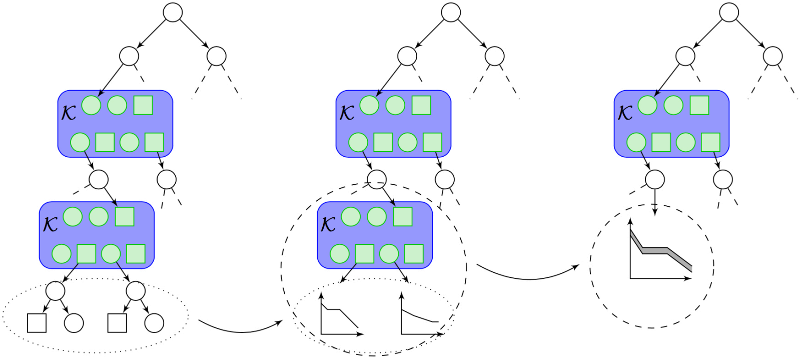
- Outside of the attractor of player $\square$ toward ✓ $\Rightarrow -\infty$
- The iterative algorithm converges on the other states in a number of steps linear with the region automaton's size, with $-\infty$ initialisation

What to do in case of undecidability?

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

What to do in case of undecidability?

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



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

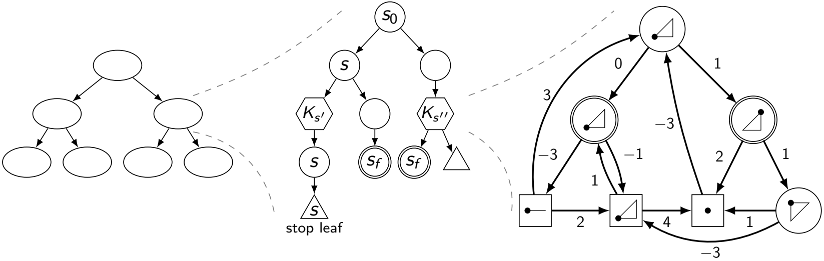
$$\text{either } (\geqslant 1 \text{ or } = 0), \qquad \text{or } (\leqslant -1 \text{ or } = 0)$$

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 ($\geqslant 1$ or $= 0$), or ($\leqslant -1$ or $= 0$)

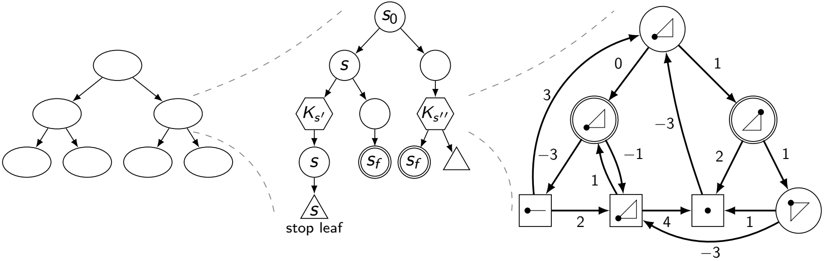


Extension in the negative case?

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

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

either ($\geqslant 1$ or $= 0$), or ($\leqslant -1$ or $= 0$)



Theorem:

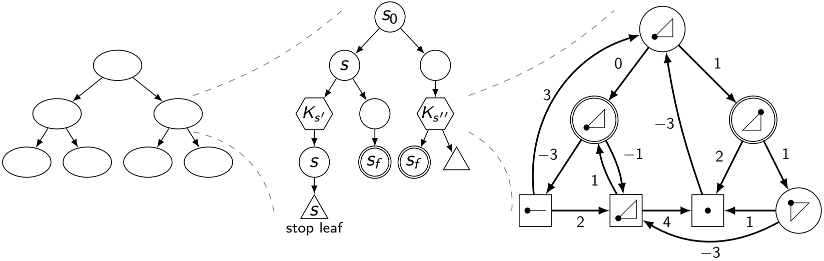
- Approximation is decidable (in doubly exponential time complexity) for almost-divergent WTGs.
- We also provide a (semi-)symbolic algorithm that does not rely on an a-priori discretisation of the regions with a fixed granularity $1/N$.

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 ($\geqslant 1$ or $= 0$), or ($\leqslant -1$ or $= 0$)



Theorem:

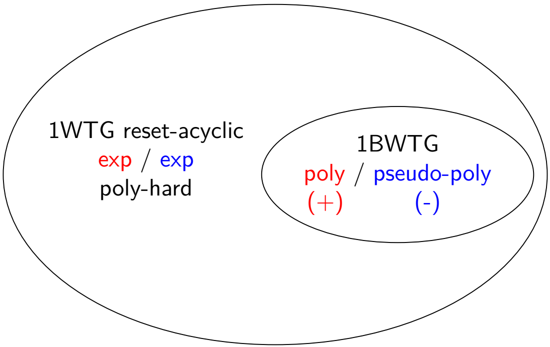
- Approximation is decidable (in doubly exponential time complexity) for almost-divergent WTGs.
- We also provide a (semi-)symbolic algorithm that does not rely on an a-priori discretisation of the regions with a fixed granularity $1/N$.

- circumvent the need for an SCC decomposition?

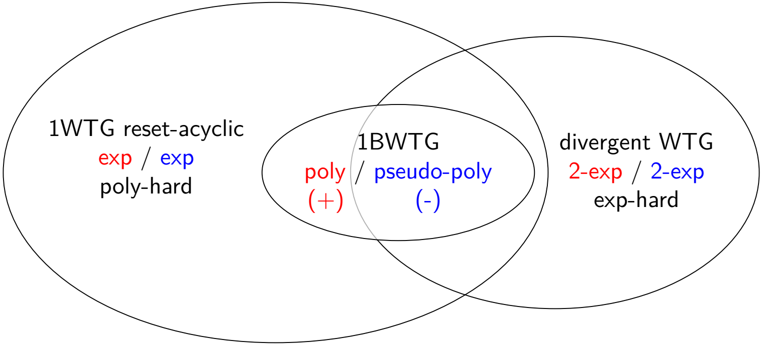
Conclusion

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

Conclusion

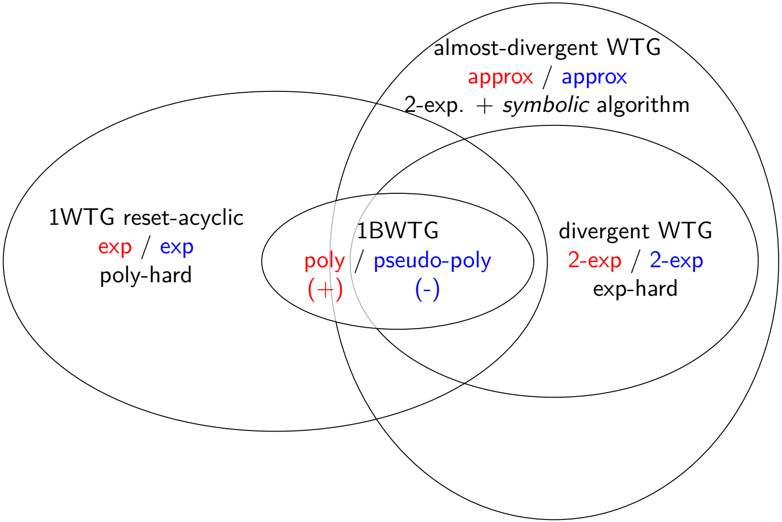


Conclusion



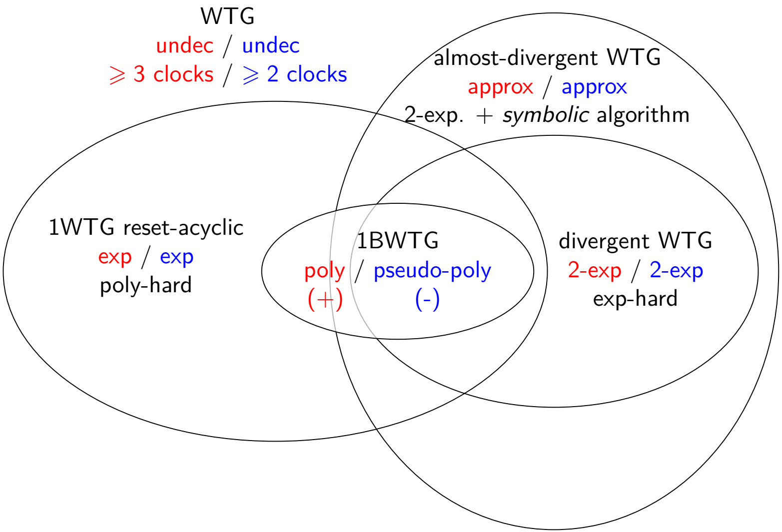
1WTG reset-acyclic
exp / exp
poly-hard
1BWTG
poly / pseudo-poly
(+) (-)
divergent WTG
2-exp / 2-exp
exp-hard

Conclusion

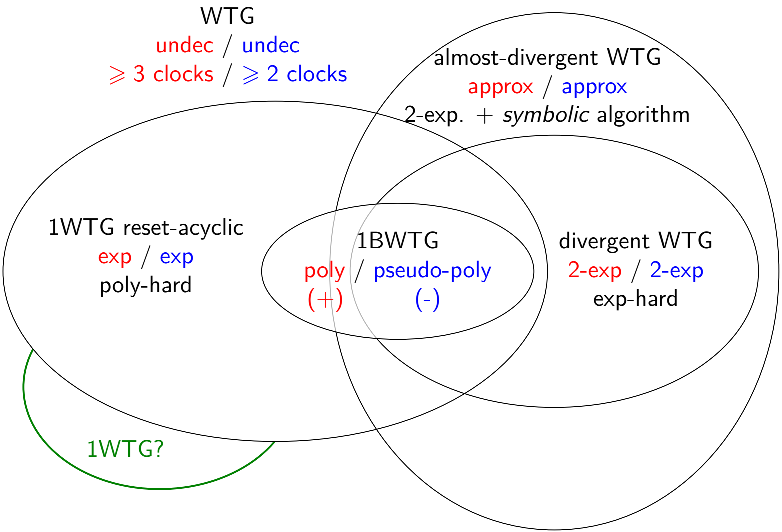


almost-divergent WTG
approx / approx
2-exp. + *symbolic* algorithm
1WTG reset-acyclic
exp / exp
poly-hard
1BWTG
poly / pseudo-poly
(+) (-)
divergent WTG
2-exp / 2-exp
exp-hard

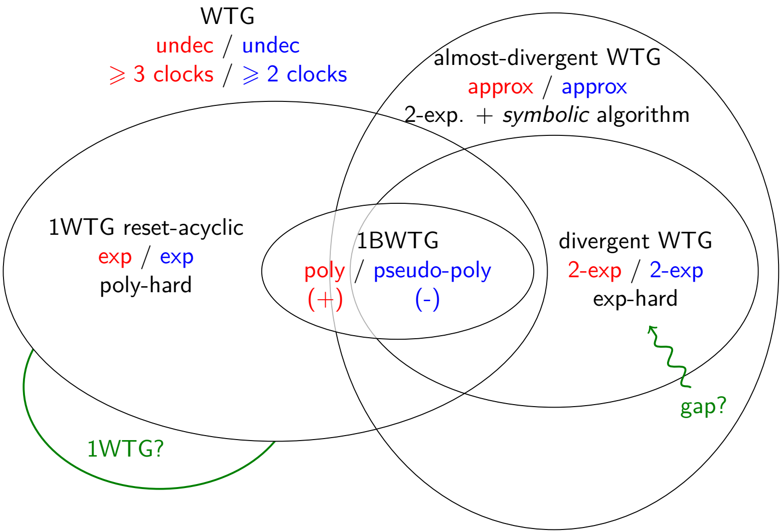
Conclusion


WTG
undec / undec
⩾ 3 clocks / ⩾ 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

Conclusion

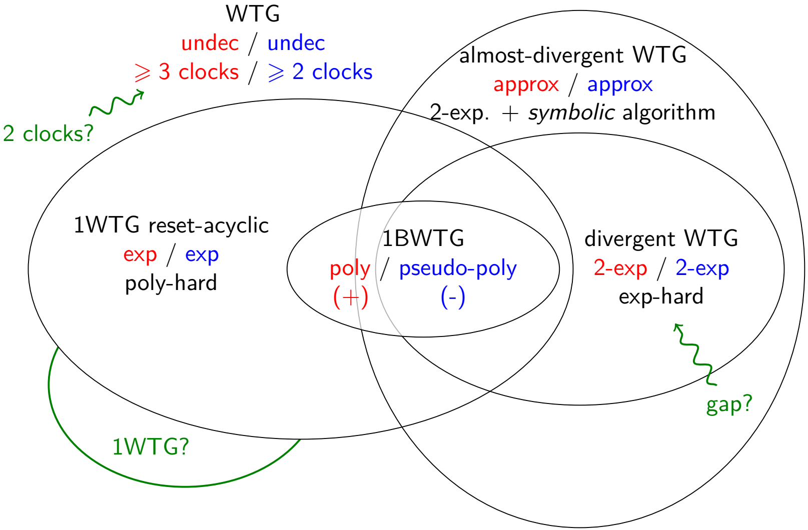

WTG
undec / undec
⩾ 3 clocks / ⩾ 2 clocks
almost-divergent WTG
approx / approx
2-exp. + *symbolic* algorithm
1WTG reset-acyclic
exp / exp
poly-hard
1BWTG
poly / pseudo-poly
(+) (-)
divergent WTG
2-exp / 2-exp
exp-hard
1WTG?

Conclusion

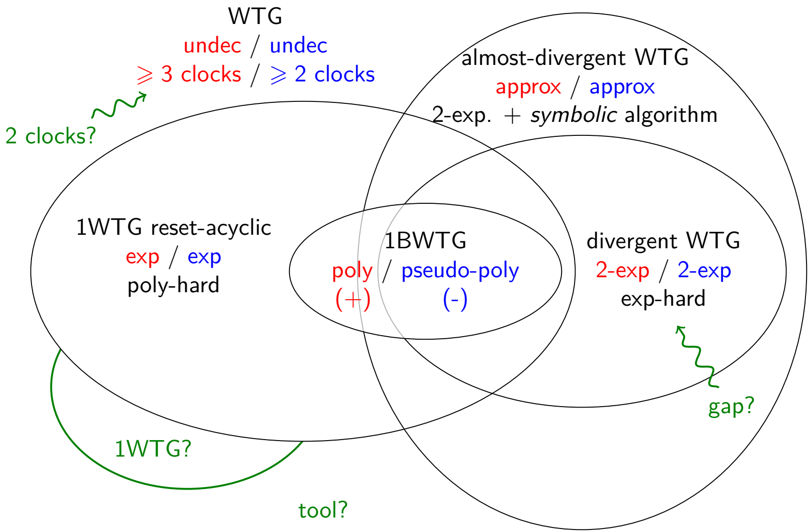


WTG
undec / undec
$\geqslant$ 3 clocks / $\geqslant$ 2 clocks
almost-divergent WTG
approx / approx
2-exp. + *symbolic* algorithm
1WTG reset-acyclic
exp / exp
poly-hard
1BWTG
poly / pseudo-poly
(+) (-)
divergent WTG
2-exp / 2-exp
exp-hard
gap?
1WTG?

Conclusion

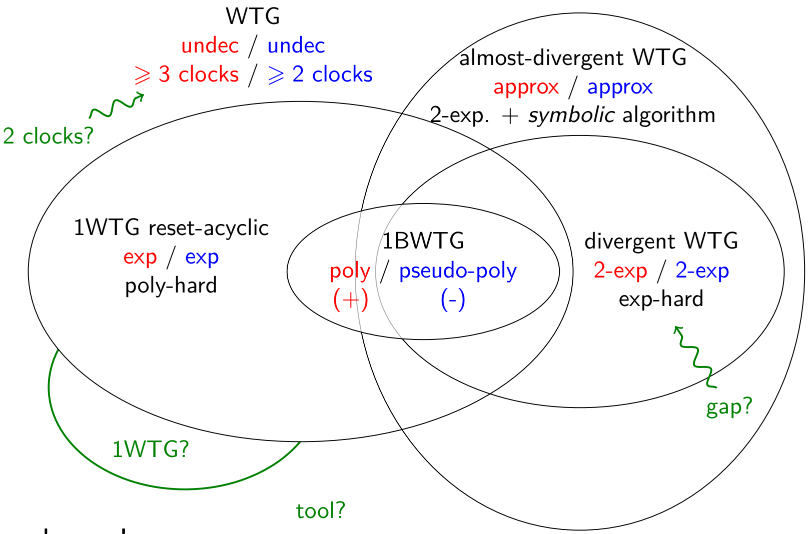

WTG
undec / undec
⩾ 3 clocks / ⩾ 2 clocks
2 clocks?
almost-divergent WTG
approx / approx
2-exp. + *symbolic* algorithm
1WTG reset-acyclic
exp / exp
poly-hard
1BWTG
poly / pseudo-poly
(+) (-)
divergent WTG
2-exp / 2-exp
exp-hard
gap?
1WTG?

Conclusion



WTG
undec / undec
⩾ 3 clocks / ⩾ 2 clocks
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?
tool?

Conclusion



Thank you!

References I

Alur, R., Bernadsky, M., and Madhusudan, P. (2004a). Optimal reachability for weighted timed games. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *Lecture Notes in Computer Science*, pages 122–133. Springer.

Alur, R., Bernadsky, M., and Madhusudan, P. (2004b). Optimal reachability for weighted timed games. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *LNCS*, pages 122–133. Springer.

Bouyer, P., Brihaye, T., Bruyère, V., and Raskin, J.-F. (2007). On the optimal reachability problem of weighted timed automata. *Formal Methods in System Design*, 31(2):135–175.

Bouyer, P., Brihaye, T., and Markey, N. (2006a). Improved undecidability results on weighted timed automata. *Information Processing Letters*, 98(5):188–194.

Bouyer, P., Brinksma, E., and Larsen, K. G. (2004a). Staying alive as cheaply as possible. In *Hybrid Systems: Computation and Control*, pages 203–218. Springer.

Bouyer, P., Cassez, F., Fleury, E., and Larsen, K. G. (2004b). Optimal strategies in priced timed game automata. In *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *Lecture Notes in Computer Science*, pages 148–160. Springer.

References II

Bouyer, P., Cassez, F., Fleury, E., and Larsen, K. G. (2004c). Optimal strategies in priced timed game automata. In *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *LNCS*, pages 148–160. Springer.

Bouyer, P., Jaziri, S., and Markey, N. (2015). On the value problem in weighted timed games. In *Proceedings of the 26th International Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *Leibniz International Proceedings in Informatics*, pages 311–324. Leibniz-Zentrum für Informatik.

Bouyer, P., Larsen, K. G., Markey, N., and Rasmussen, J. I. (2006b). Almost optimal strategies in one-clock priced timed games. In *Proceedings of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 345–356. Springer.

Brihaye, T., Bruyère, V., and Raskin, J.-F. (2005). On optimal timed strategies. In *Proceedings of the Third international conference on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *Lecture Notes in Computer Science*, pages 49–64. Springer.

Brihaye, T., Geeraerts, G., Haddad, A., Lefaucheux, E., and Monmege, B. (2015). Simple priced timed games are not that simple. In *Proceedings of the 35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'15)*, volume 45 of *LIPIcs*, pages 278–292. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

References III

Brihaye, T., Geeraerts, G., Haddad, A., and Monmege, B. (2016). Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Informatica*.

Brihaye, T., Geeraerts, G., Narayanan Krishna, S., Manasa, L., Monmege, B., and Trivedi, A. (2014). Adding negative prices to priced timed games. In *Proceedings of the 25th International Conference on Concurrency Theory (CONCUR'14)*, volume 8704, pages 560–575. Springer.

Busatto-Gaston, D., Monmege, B., and Reynier, P.-A. (2017). Optimal reachability in divergent weighted timed games. In Esparza, J. and Murawski, A. S., editors, *Proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'17)*, volume 10203 of *Lecture Notes in Computer Science*, pages 162–178, Uppsala, Sweden. Springer.

Fearnley, J. and Jurdziński, M. (2013). Reachability in two-clock timed automata is PSPACE-complete. In *Proceedings of ICALP'13*, volume 7966 of *Lecture Notes in Computer Science*, pages 212–223. Springer.

Haase, C., Ouaknine, J., and Worrell, J. (2012). On the relationship between reachability problems in timed and counter automata. In *Proceedings of RP'12*, pages 54–65.

Hansen, T. D., Ibsen-Jensen, R., and Miltersen, P. B. (2013). A faster algorithm for solving one-clock priced timed games. In *Proceedings of the 24th International Conference on Concurrency Theory (CONCUR'13)*, volume 8052 of *LNCS*, pages 531–545. Springer.

References IV

Khachiyan, L., Boros, E., Borys, K., Elbassioni, K., Gurvich, V., Rudolf, G., and Zhao, J. (2008). On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233.

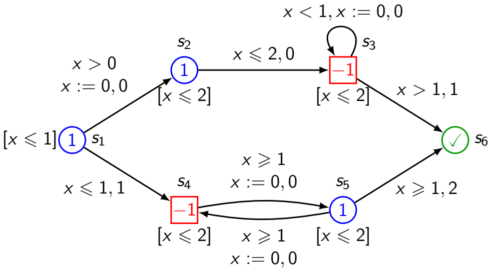
Rutkowski, M. (2011). Two-player reachability-price games on single-clock timed automata. In *Proceedings of the Ninth Workshop on Quantitative Aspects of Programming Languages (QAPL'11)*, volume 57 of *Electronic Proceedings in Theoretical Computer Science*, pages 31–46.

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



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

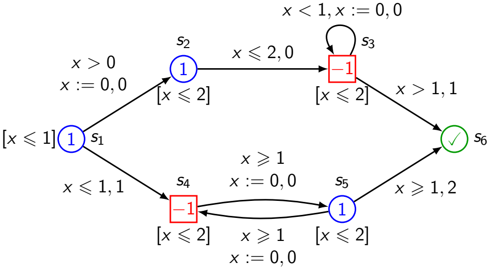
Player ○ wants to leave as soon as possible a state with rate p^+, and wants to stay as long as possible in a state with rate p^-: so, he will always play η-close to a border...

Lemma:

Both players can play arbitrarily close to borders w.l.o.g.: for every η

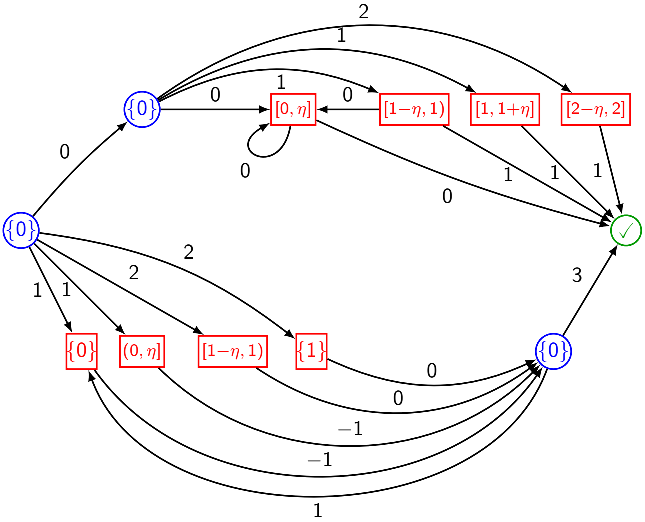
$$\underline{\mathsf{Val}}^{\eta}(s, v) \leqslant \underline{\mathsf{Val}}(s, v) \quad \leqslant \quad \overline{\mathsf{Val}}(s, v) \leqslant \overline{\mathsf{Val}}^{\eta}(s, v)$$

2. Finite weighted game abstraction



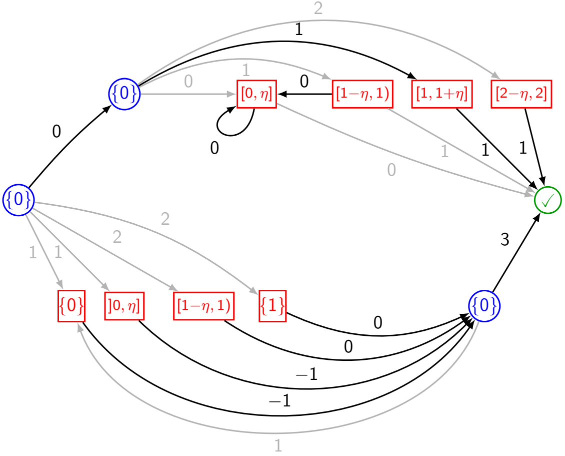
η-regions: $\{0\}, (0, \eta), (1-\eta, 1), \{1\}, (1, 1+\eta), (2-\eta, 2), \{2\}, (2, +\infty)$

2. Finite weighted game abstraction



3. Study $\mathcal{G}$: values, optimal strategies of a min-cost reachability game

(Brihaye et al., 2016)



Optimal value: $\mathsf{Val}_{\mathcal{G}}(s_1, \{0\}) = +2$ (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:

$$\mathsf{Val}_{\mathcal{G}}(s, \{0\}) - \varepsilon \leqslant \underline{\mathsf{Val}}^{\eta}(s, 0) \leqslant \underline{\mathsf{Val}}(s, 0) \leqslant \overline{\mathsf{Val}}(s, 0) \leqslant \overline{\mathsf{Val}}^{\eta}(s, 0) \leqslant \mathsf{Val}_{\mathcal{G}}(s, \{0\}) + \varepsilon$$

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:

$$\mathsf{Val}_{\mathcal{G}}(s,\{0\}) - \varepsilon \leqslant \underline{\mathsf{Val}}^{\eta}(s,0) \leqslant \underline{\mathsf{Val}}(s,0) \leqslant \overline{\mathsf{Val}}(s,0) \leqslant \overline{\mathsf{Val}}^{\eta}(s,0) \leqslant \mathsf{Val}_{\mathcal{G}}(s,\{0\}) + \varepsilon$$

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