Reach and get rich:
pseudo-polynomial algorithms for total-payoff games

Thomas Brihaye[1], Gilles Geeraerts[2], Axel Haddad[1] (**me**), Benjamin Monmege[2]

[1]Université de Mons
[2]Université de Bruxelles
European Project FP7-CASSTING

Teaser

- **Variant** of usual quantitative games
- Add a **reachability objective**
- We want to **compute** the value
- Game extension of **shortest path** problem
- Solve an **open problem** for total-payoff games

2-player quantitative games on graph

Eve plays against **Adam**. The **arena** is:

- a finite **graph**,
- where the vertices belong either to **Eve** or **Adam**,
- and each edge has a **weight**.

During a **play**:

- A **token** is moved along the edges
- by the player that owns the current state.
- The play is **infinite**.

Payoff function

Defines a **value** of a play.

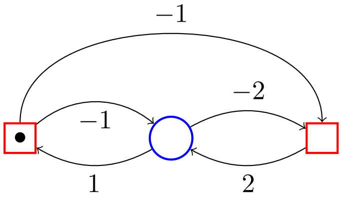
Total Payoff: the limit of the **sums** of the weigths.

Mean Payoff: the limit of the **average** of the weights.

(actually we take the limit inferior)

Eve wants to **minimize** it, **Adam** wants to **maximize** it.

Example

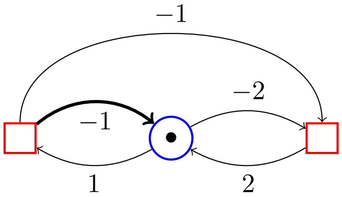


Weights:

Sums:

Average:

Example

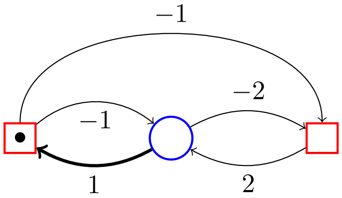


Weights: -1

Sums: -1

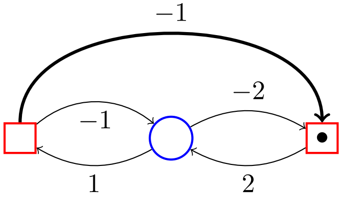
Average: -1

Example



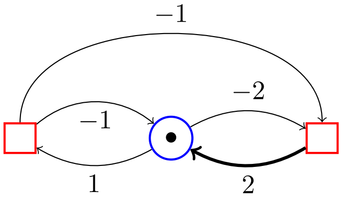
Weights:	-1	1
Sums:	-1	0
Average:	-1	0

Example



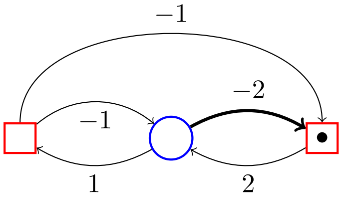
Weights:	-1	1	-1
Sums:	-1	0	-1
Average:	-1	0	-0.333

Example



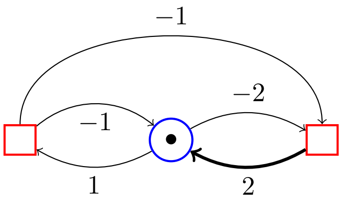
Weights:	-1	1	-1	2
Sums:	-1	0	-1	1
Average:	-1	0	-0.333	0.25

Example



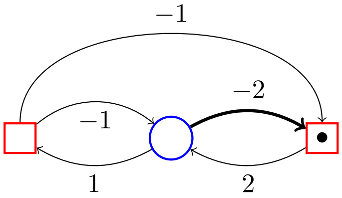
Weights:	-1	1	-1	2	-2
Sums:	-1	0	-1	1	-1
Average:	-1	0	-0.333	0.25	-0.2

Example



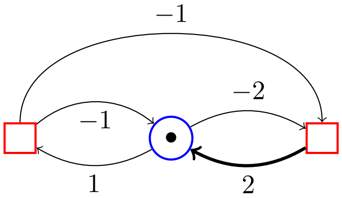
Weights:	-1	1	-1	2	-2	2
Sums:	-1	0	-1	1	-1	1
Average:	-1	0	-0.333	0.25	-0.2	0.166

Example



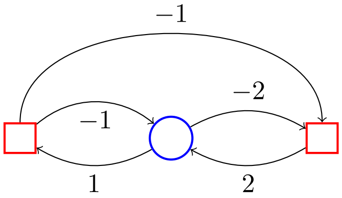
Weights:	-1	1	-1	2	-2	2	-2
Sums:	-1	0	-1	1	-1	1	-1
Average:	-1	0	-0.333	0.25	-0.2	0.166	-0.143

Example



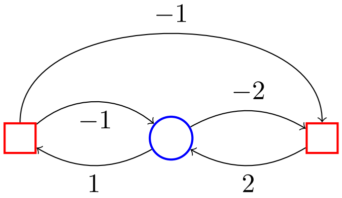
Weights:	-1	1	-1	2	-2	2	-2	2
Sums:	-1	0	-1	1	-1	1	-1	1
Average:	-1	0	-0.333	0.25	-0.2	0.166	-0.143	0.125

Example



Weights:	-1	1	-1	2	-2	2	-2	2	$\cdots$
Sums:	-1	0	-1	1	-1	1	-1	1	$\cdots$
Average:	-1	0	-0.333	0.25	-0.2	0.166	-0.143	0.125	$\cdots$

Example



Weights:	-1	1	-1	2	-2	2	-2	2	$\cdots$
Sums:	-1	0	-1	1	-1	1	-1	1	$\cdots$
Average:	-1	0	-0.333	0.25	-0.2	0.166	-0.143	0.125	$\cdots$

Total Payoff: -1 **Mean Payoff:** 0

Strategies

Strategie: given the **past**, what choice to make.

Value of the strategie σ from vertex v:

- For **Eve**: **supremum** of the values of the plays in $\mathsf{Play}(v, \sigma)$,
- For **Adam**: **infimum** of the values of the plays in $\mathsf{Play}(v, \sigma)$,

(*i.e.*, the worst thing that can happen to me)

Value of a vertex v:

- For **Eve**: **infimum** value over **her** strategies.
- For **Adam**: **supremum** value over **his** strategies.

(*i.e.*, the best thing that I can do)

Determinacy. The **Eve**-value and the **Adam**-value of any vertex v are **equal**. [consequence of Martin 75]

Well-known results

Positional strategy: strategy that depends only on the **current node**.

There exists **optimal positional strategies** for both players [Ehrenfeucht, Mycielski 79] [Gimbert, Zielonka 04].

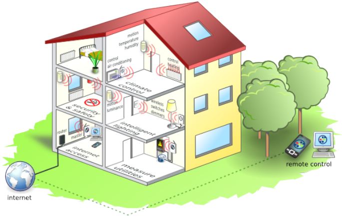
Deciding whether the value of a vertex is $\leqslant K$ is in $\mathbf{NP} \cap \mathbf{coNP}$ (no known algorithm in $\mathbf{P}$).

For **Mean Payoff** one can compute the values in **pseudo-polynomial time** [Zwick, Paterson 95].

Our motivation

Cassting

> “The objective is to develop a novel approach for analysing and designing collective adaptive systems in their totality, by setting up a game theoretic framework.”



Priced Timed Games $\xrightarrow{\text{models?}}$ Small energy trading network

Priced Timed Games $\longrightarrow$ **Reachability Quantitative Games**

Reachability quantitative games

Take:

- an **arena**,
- some **target vertices** **T**,
- a payoff function **P**.

Introduce a new payoff function **T**-**RP**. The value of a play π is:

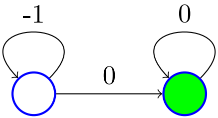
- If π does not reach a target: **T**-**RP**$(\pi) = +\infty$
- If π reaches a target $\pi = v_1 \cdots v_k$ **t** $v_{k+2} \cdots$:

$$\mathsf{T}-\mathsf{RP}(\pi) = \mathsf{P}(v_1 \cdots v_k).$$

Eve wants to **reach** a **target** while **minimizing** the payoff.

Adam wants to **avoid** the **target** or **maximize** the payoff.

Example



$\mathsf{Val} = -\infty$ **but** no optimal strategy!

What is known

These game are **determined** [consequence of Martin 75].

Best strategies are of the form:

- play **for a long time** a **positional strategy**
- and then **reach** the target

[Filiot, Gentilini, Raskin 12].

Deciding whether the value of a vertex is $\leqslant K$ is in $\mathbf{NP} \cap \mathbf{coNP}$.

Total Payoff, Non-negative weights. In this case, **positionally determined**, value and optimal strategies **can be computed** in $\mathbf{P}$ (modified Dijkstra algorithm) [Kachiyan et Al. 08].

Contributions

Reachability mean-payoff games are equivalent to **mean-payoff game**.

$\Rightarrow$ One can compute the values in pseudo-polynomial time.

A **value iteration** algorithm for **reachability total-payoff** games:

$\Rightarrow$ it computes the values in pseudo-polynomial time.

A **value iteration** algorithm for **total-payoff** games (also pseudo-polynomial).

Computing the attractor

Attractor: all the vertices from which **Eve** has a strategy to reach the **targets**.

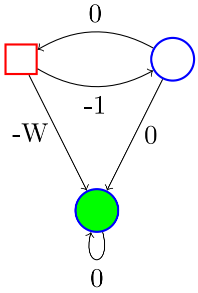
Eve ensures a value $< +\infty$ **if and only if** the plays never leave the **attractor**.

Computing the attractor is in $\mathbf{P}$.

$\Rightarrow$ We always assume that we have **removed** all the vertices **not in the attractor**.

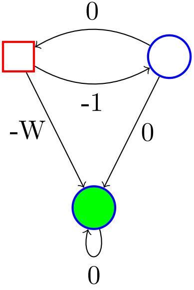
An example

Even when **optimal** strategies exists, they might need memory!



An example

Even when **optimal** strategies exists, they might need memory!



Optimal strategy for **Eve**: go $\leftarrow$ W times and then go $\downarrow$
Optimal strategy for **Adam**: go $\downarrow$

Value iteration for RTP games

Compute $\mathsf{Val}^{\leqslant i}$ the value mapping when the game stops after i steps.

Value iteration for RTP games

Compute $\mathsf{Val}^{\leqslant i}$ the value mapping when the game stops after i steps.

$\mathsf{Val}^{\leqslant 0}$ = Everything is $+\infty$, or 0 for the **targets** (the **greatest** possible value function)

$\mathsf{Val}^{\leqslant i+1} = \mathcal{F}(\mathsf{Val}^{\leqslant i})$ with $\mathcal{F}$ a continuous monotonic function.

$\Rightarrow$ the sequence converges towards the **greatest fixpoint**,

Value iteration for **RTP** games

Compute $\mathsf{Val}^{\leqslant i}$ the value mapping when the game stops after i steps.

$\mathsf{Val}^{\leqslant 0}$ = Everything is $+\infty$, or 0 for the **targets** (the **greatest** possible value function)

$\mathsf{Val}^{\leqslant i+1} = \mathcal{F}(\mathsf{Val}^{\leqslant i})$ with $\mathcal{F}$ a continuous monotonic function.

$\Rightarrow$ the sequence converges towards the **greatest fixpoint**,

Furthermore, the sequence stabilises (or value $-\infty$ is detected) after at most a **pseudo-polynomial** number of steps.

Value iteration for RTP games

Compute $\mathsf{Val}^{\leqslant i}$ the value mapping when the game stops after i steps.

$\mathsf{Val}^{\leqslant 0}$ = Everything is $+\infty$, or 0 for the **targets** (the **greatest** possible value function)

$\mathsf{Val}^{\leqslant i+1} = \mathcal{F}(\mathsf{Val}^{\leqslant i})$ with $\mathcal{F}$ a continuous monotonic function.

$\Rightarrow$ the sequence converges towards the **greatest fixpoint**,

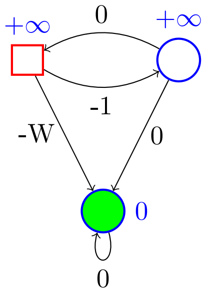
Furthermore, the sequence stabilises (or value $-\infty$ is detected) after at most a **pseudo-polynomial** number of steps.

This **greatest fixpoint** is equal to Val.

$\Rightarrow$ A **pseudo-polynomial** algorithm for computing the value!

Running the algorithm

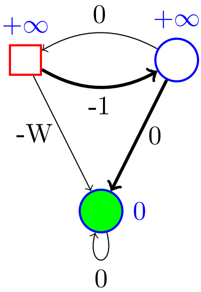
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$+\infty$	$+\infty$	0

Running the algorithm

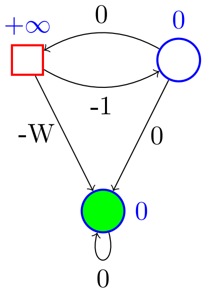
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



		
$+\infty$	$+\infty$	0

Running the algorithm

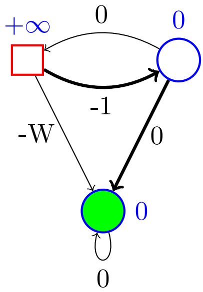
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$\square$	$\bigcirc$	$\bullet$
$+\infty$	$+\infty$	0
$+\infty$	0	0

Running the algorithm

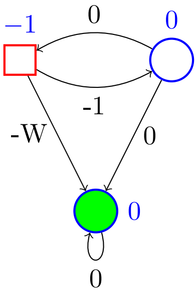
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$+\infty$	$+\infty$	0
$+\infty$	0	0

Running the algorithm

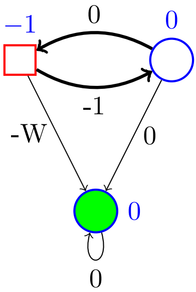
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0

Running the algorithm

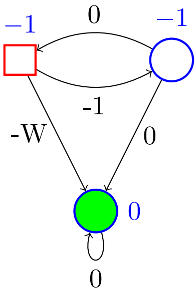
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0

Running the algorithm

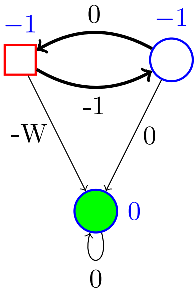
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



□	○	●
$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0
-1	-1	0

Running the algorithm

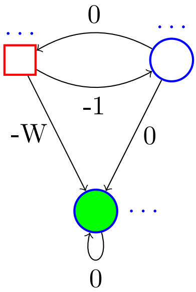
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



□	○	●
$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0
-1	-1	0

Running the algorithm

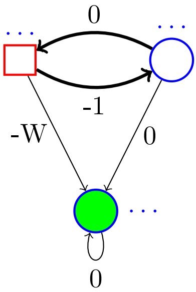
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$\square$	$\bigcirc$	$\bullet$
$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0
-1	-1	0
$\vdots$	$\vdots$	$\vdots$

Running the algorithm

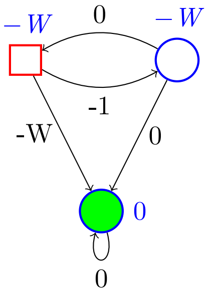
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0
-1	-1	0
$\vdots$	$\vdots$	$\vdots$

Running the algorithm

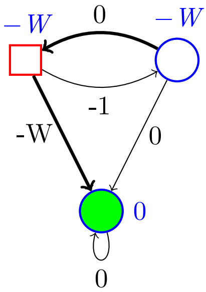
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0
-1	-1	0
$\vdots$	$\vdots$	$\vdots$
$-W$	$-W$	0

Running the algorithm

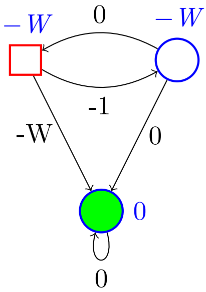
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0
-1	-1	0
$\vdots$	$\vdots$	$\vdots$
$-W$	$-W$	0

Running the algorithm

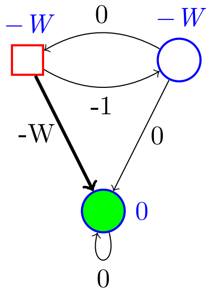
$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0
-1	-1	0
$\vdots$	$\vdots$	$\vdots$
$-W$	$-W$	0
$-W$	$-W$	0

Running the algorithm

$\mathsf{Val}^{\leqslant i+1}$ = do **one move**, and get the values of $\mathsf{Val}^{\leqslant i}$.



□	○	●
$+\infty$	$+\infty$	0
$+\infty$	0	0
-1	0	0
-1	-1	0
$\vdots$	$\vdots$	$\vdots$
$-W$	$-W$	0
$-W$	$-W$	0

optimal positional strategy for **Adam**

Back to classical total-payoff games

No known efficient algorithm for computing the value of total-payoff games (without the reachability condition).

We use **reachability total-payoff** games to solve **total-payoff** games.

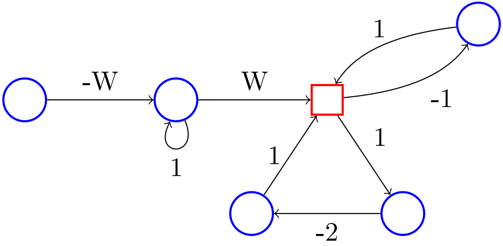
Introduce a **pseudo-polynomial time** transformation from **TP** to **RTP**

$\Rightarrow$ A pseudo-polynomial time iteration algorithm for computing the value of total-payoff games.

How does it work

(Recall that $\textbf{TP}(v_1 v_2 \cdots) = \lim\textbf{inf}\ \mathrm{Sum}(v_1 \cdots v_i)$)

- At each step **Eve** can ask to stop the game,
- **Adam** can refuse **K** times,
- **K** is pseudo-polynomial (here take $\mathbf{K} = W + 2$).



- This can be encoded in a pseudo-polynomial size **RTP** game.

(**actually**, we do not need to compute the whole game)

Conclusion

- **Reachability mean-payoff games** are equivalent to **mean-payoff games** (pseudo-polynomial algorithm)
- **Value iteration** algorithm for **reachability total-payoff games** (pseudo-polynomial algorithm)
- **Value iteration** algorithm for **total-payoff games** (pseudo-polynomial algorithm)
- **More:** Acceleration
- **More:** Finding good strategies for **Eve** and **Adam** in **RTP** games and in **TP** games.
- **Thanks! ... Questions?**