To reach or not to reach?
Efficient algorithms for total-payoff games

Thomas Brihaye\textsuperscript{1}, Gilles Geeraerts\textsuperscript{2}, Axel Haddad\textsuperscript{1} (me), Benjamin Monmege\textsuperscript{2}

\textsuperscript{1}Université de Mons
\textsuperscript{2}Université de Bruxelles
European Project FP7-CASSTING
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
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 weights.

**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:

Total Payoff: 1
Mean Payoff: 0
Example

Weights: -1

Sums: -1

Average: -1
Example

Weights: \(-1\) \hspace{1em} 1

Sums: \(-1\) \hspace{1em} 0

Average: \(-1\) \hspace{1em} 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:

\[
\begin{array}{cccccc}
-1 & 1 & -1 & 2 & -2 \\
\end{array}
\]

Sums:

\[
\begin{array}{cccccc}
-1 & 0 & -1 & 1 & -1 \\
\end{array}
\]

Average:

\[
\begin{array}{cccccc}
-1 & 0 & -0.333 & 0.25 & -0.2 \\
\end{array}
\]
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  ... 

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:  \(-1\)  \(0\)  \(-0.333\)  \(0.25\)  \(-0.2\)  \(0.166\)  \(-0.143\)  \(0.125\)  \(\cdots\)

Total Payoff:  \(-1\)  \(\text{Mean Payoff:}\ 0\)
Known results

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

(Positional strategy = strategy that depends only on the current node)

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

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

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

(\(\text{Eve} \) gets \(+\infty\) if she does not reach a **target**)

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

![Diagram showing the game setup and possible strategies for Eve and Adam.](image-url)
Reachability quantitative games

Add some target vertices.

Eve wants to reach a target while minimizing the payoff.

(Eve gets $+\infty$ if she does not reach a target)

Adam wants to avoid the target or maximize the payoff.

Val $= -\infty$ but no optimal strategy!
Reachability quantitative games

Add some target vertices.

Eve wants to reach a target while minimizing the payoff.

(Eve gets $+\infty$ if she does not reach a target)

Adam wants to avoid the target or maximize the payoff.

Val = $-\infty$ but no optimal strategy!

Optimal strategy for Eve:

go $\leftarrow$ W times and then go $\downarrow$

Optimal strategy for Adam: go $\downarrow$
What is known

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 \( \leq K \) is in \( \text{NP} \cap \text{coNP} \).

Total Payoff, Non-negative weights. In this case, positionally determined, value and optimal strategies can be computed in \( \text{P} \) (modified Dijkstra algorithm) [Kachiyan et Al. 08].
Reachability mean-payoff games are equivalent to mean-payoff games.
⇒ One can compute the values in pseudo-polynomial time.

A value iteration algorithm for reachability total-payoff games:
⇒ it computes the values in pseudo-polynomial time.

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

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

$$(\text{Val}^{\leq i+1} = \text{do one move}, \text{and get the values of } \text{Val}^{\leq i})$$
Algorithm for **reachability total-payoff**

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

$$(Val^{\leq i+1} = \text{do one move}, \text{ and get the values of } Val^{\leq i})$$
Algorithm for **reachability total-payoff**

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

$(\text{Val}^{\leq i+1} = \text{do one move}, \text{and get the values of } \text{Val}^{\leq i})$
Algorithm for reachability total-payoff

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

($\text{Val}^{\leq i+1} = \text{do one move}$, and get the values of $\text{Val}^{\leq i}$)
Algorithm for **reachability total-payoff**

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

($\text{Val}^{\leq i+1} = \text{do one move}$, and get the values of $\text{Val}^{\leq i}$)
Algorithm for **reachability total-payoff**

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

$(Val^{\leq i+1} = \text{do one move}, \text{ and get the values of } Val^{\leq i})$
Algorithm for **reachability total-payoff**

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

$(\text{Val}^{\leq i+1} = \text{do one move}, \text{and get the values of Val}^{\leq i})$
Algorithm for reachability total-payoff

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

$(\text{Val}^{\leq i+1} = \text{do one move}, \text{and get the values of Val}^{\leq i})$
Algorithm for **reachability total-payoff**

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

$$(\text{Val}^{\leq i+1} = \text{do one move}, \text{and get the values of } \text{Val}^{\leq i})$$
Algorithm for **reachability total-payoff**

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

$$(\text{Val}^{\leq i+1} = \text{do one move}, \text{and get the values of Val}^{\leq i})$$
Algorithm for **reachability total-payoff**

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

($\text{Val}^{\leq i+1} = \text{do one move},$ and get the values of $\text{Val}^{\leq i}$)

```
<table>
<thead>
<tr>
<th></th>
<th>( )</th>
<th>( )</th>
<th>( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>+∞</td>
<td>+∞</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>+∞</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>−1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>−1</td>
<td>−1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td></td>
</tr>
<tr>
<td>−$W$</td>
<td>−$W$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```
Algorithm for reachability total-payoff

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

$(\text{Val}^{\leq i+1} = \text{do one move}, \text{and get the values of } \text{Val}^{\leq i})$
Algorithm for **reachability total-payoff**

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

$(\text{Val}^{\leq i+1} = \text{do one move}, \text{ and get the values of } \text{Val}^{\leq i})$
Algorithm for **reachability total-payoff**

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

$$(\text{Val}^{\leq i+1} = \text{do one move}, \text{and get the values of } \text{Val}^{\leq i})$$

![Game diagram](image)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;+∞&amp;+∞&amp;0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;+∞&amp;0&amp;0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;−1&amp;0&amp;0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;−1&amp;−1&amp;0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;⋯&amp;⋯&amp;⋯</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;−$W$&amp;−$W$&amp;0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;−$W$&amp;−$W$&amp;0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

optimal positional strategy for **Adam**
Algorithm for total-payoff

construct a RTP game

compute the values

update the game

... until it converges ...

A: -W, B: 1, C: 0

A: -W + 1, B: 2, C: 0

A: -0, B: W, C: 0
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?**