A pseudo-quasi-polynomial algorithm for solving mean-payoff parity games

Laure Daviaud
University of Warwick

Joint work with Marcin Jurdziński and Ranko Lazić
Labri, Bordeaux, 05/06/2018
Mean-payoff Parity Games

- Two players, zero-sum
- Ensure a parity condition while minimising some cost
Two players - Triangle and Rectangle
Parity Game - Odd and Even

The limsup of the priorities has to be odd
The limsup of the priorities has to be odd

\[ \implies \text{Odd wins from everywhere} \]
Mean-payoff Parity Games

Mean payoff Game - Min and Max

\[ \text{Limsup of the average sums has to be negative} \]
Mean-payoff Parity Games

Mean payoff Game - Min and Max

\[ \text{Lim sup of the average sums has to be negative} \rightarrow \text{Min wins from everywhere} \]
Mean-payoff Parity Games

→ Con has to ensure that both the parity and the mean payoff conditions are satisfied: Odd \& Negative
Mean-payoff Parity Games

Mean-payoff Parity Game - Con and Dis

Con has to ensure that both the parity and the mean payoff conditions are satisfied: Odd ∧ Negative

⇒ Dis wins from everywhere!
Mean-Payoff Parity Games

Introduced by Chatterjee, Henziger, Jurdziński

Combine qualitative and quantitative objectives

Is there a winner? Who? Compute strategies...

Efficient algorithm to solve the game
- Holy Grail: polynomial time -
Determinacy and positionality

- Determinacy
- Positional strategy for Dis

![Diagram showing positional strategy for Dis]
Determinacy and positionality

- Determinacy
- Positional strategy for \( \text{Dis} \)
A bit of history

Mean-Payoff Games

- Positional determinacy [Ehrenfeuch&Mycielski]
- Pseudo-polynomial algorithms - $O(n^k C)$
  [Paterson&Zwick, Brim&Chaloupka&Doyen&Gentilini&Raskin, Comin&Rizzi...]

Our result: Pseudo-quasi polynomial algorithm - $O(n \log(d) + k C)$
  [Chatterjee, Henzinger, Doyen, Bouyer, Markey, Olschweski, Ummels, Svozil]
A bit of history

Mean-Payoff Games

- Positional determinacy [Ehrenfeucht & Mycielski]
- Pseudo-polynomial algorithms - $O(n^k C)$
  [Paterson & Zwick, Brim & Chaloupka & Doyen & Gentilini & Raskin, Comin & Rizzi...]

Parity Games

- Recursive algorithms - $O(n^{d+k})$
  [McNaughton, Zielonka]
- Quasi-polynomial algorithms - $O(n^{\log(d)+k})$
  [Calude & Jain & Khoussainov & Li & Stephan, Jurdziński & Lazić,...]
A bit of history

Mean-Payoff Games

- Positional determinacy [Ehrenfeucht&Mycielski]
- Pseudo-polynomial algorithms - $O(n^k C)$
  [Paterson&Zwick, Brim&Chaloupka&Doyen&Gentilini&Raskin, Comin&Rizzi...]

Parity Games

- Recursive algorithms - $O(n^{d+k})$
  [McNaughton, Zielonka]
- Quasi-polynomial algorithms - $O(n^{\log(d)+k})$
  [Calude&Jain&Khoussainov&Li&Stephan, Jurdziński&Lazić,...]

Mean-Payoff Parity Games

- $O(n^{d+k} C)$
  [Chatterjee, Henzinger, Doyen, Bouyer, Markey, Olschweski, Ummels, Svozil]
A bit of history

Mean-Payoff Games

- Positional determinacy [Ehrenfeucht&Mycielski]
- Pseudo-polynomial algorithms - $O(n^k C)$
  [Paterson&Zwick, Brim&Chaloupka&Doyen&Gentilini&Raskin, Comin&Rizzi...]

Parity Games

- Recursive algorithms - $O(n^{d+k})$
  [McNaughton, Zielonka]
- Quasi-polynomial algorithms - $O(n^{\log(d)+k})$
  [Calude&Jain&Khoussainov&Li&Stephan, Jurdziński&Lazić,...]

Mean-Payoff Parity Games

- $O(n^{d+k} C)$
  [Chatterjee, Henzinger, Doyen, Bouyer, Markey, Olschweski, Ummels, Svozil]

Our result: Pseudo-quasi polynomial algorithm - $O(n^{\log(d)+k} C)$
Strategy decomposition - if Dis is winning

\[ \text{Dis} : \text{Even } v \geq 0 \]
Strategy decomposition - if Dis is winning

Dis: Even \( v \geq 0 \)

\( b \): highest priority even

Recursive decomposition

Attractor for Dis

Vertices of priority \( b \) even
Strategy decomposition - if Dis is winning

\[ \text{Dis : Even } v \geq 0 \]

\[ b : \text{highest priority even} \]

\[ b : \text{highest priority odd} \]
Strategy decomposition - if Dis is winning

$\text{Con} : \text{Odd} \land < 0$
Strategy decomposition - if Dis is winning

Con: Odd ∧ < 0

b: highest priority odd

Recursive decomposition

Attractor for Con

Vertices of priority b odd

+ a mean payoff strategy for Con on the whole game
Strategy decomposition - if Dis is winning

\[ \text{Con: Odd} \wedge <0 \]

- If Dis is winning, consider the strategy decomposition based on priority:
  - Odd priority: \( b \), Recursive decomposition
  - Even priority: \( b \), Recursive decomposition

\[ + \text{ a mean payoff strategy for Con on the whole game} \]

- Attractor for Con
- Vertices of priority \( b \) odd
- Trap for Dis
  - Recursive decomp
Map each vertex from the game into a node in the tree.
An edge $u \to v$ is **progressive** if (with $u$ mapped to the pink node):

- $v$ is mapped to a node in the blue area
- $v$ is mapped to a node in the green area and $u$ has an even priority
The lifting algorithm for Even

Idea: move the vertices in the tree until Even can always choose a progressive edge and Odd has no other choice than choosing a progressive edge.
The lifting algorithm for Even

Idea: move the vertices in the tree until Even can always choose a progressive edge and Odd has no other choice than choosing a progressive edge

- start with all the vertices mapped to the smallest nodes
- if a vertex owned by Even has no outgoing progressive edge, move the vertex in the tree to the smallest node which makes one of the edge progressive
- if a vertex owned by Odd has at least one outgoing edge not progressive, move the vertex in the tree to the smallest node which makes all the edges progressive
- if there is no such node for a vertex, assign $\top$ to it
The lifting algorithm for Even

Idea: move the vertices in the tree until Even can always choose a progressive edge and Odd has no other choice than choosing a progressive edge

- start with all the vertices mapped to the smallest nodes
- if a vertex owned by Even has no outgoing progressive edge, move the vertex in the tree to the smallest node which makes one of the edge progressive
- if a vertex owned by Odd has at least one outgoing edge not progressive, move the vertex in the tree to the smallest node which makes all the edges progressive
- if there is no such node for a vertex, assign $\top$ to it

There exists trees such that the winning set for Odd is exactly those vertices that are assigned $\top$ by the lifting algorithm (Small progress measure [Jurdzinski], Succinct progress measure [Jurdzinski-Lazic])
Solving mean-payoff games - for Max
Solving mean-payoff games - for Max

![Game Graph]

-2 3 5 2 0 -7 5 -8 0
Solving mean-payoff games - for Max
Solving mean-payoff games - for Max
Solving mean-payoff games - for Max
Solving mean-payoff games - for Max
Solving mean-payoff games - for Max
Solving mean-payoff games - for Max
Solving mean-payoff games - for Max
Combining those two ideas into a new progress measure

Solving mean-payoff parity games
Combining those two ideas into a new progress measure
Map each vertex from the game into a node in the tree and an integer (or $\infty$)
Combining those two ideas into a new progress measure
Map each vertex from the game into a node in the tree and an integer (or $\infty$)

Lifting algorithm
One technique

Convert the strategy decomposition into a progress measure
New kind of lifting algorithm

Theorem

Depending on the asymptotic growth of $d$ as a function of $n$, the running time of the algorithm is as follows:

1. $O\left( mt^2 + o(1) \right)$ if $d = o(\log n)$;
2. $O\left( m t \log(\delta+1) + \log(e\delta) + 2 C \cdot \log d \cdot \sqrt{\log n} \right)$ if $d \leq 2 \lceil \delta \log n \rceil$, for some positive constant $\delta$;
3. $O\left( dm n \log(d/\log n) + 2.45 C \right)$ if $d = \omega(\log n)$.

The algorithm works in space $O(n \cdot \log n \cdot \log d)$.