Algorithmic Game Theory and Applications
Lecture 13: Parity Games and Mean-Payoff Games

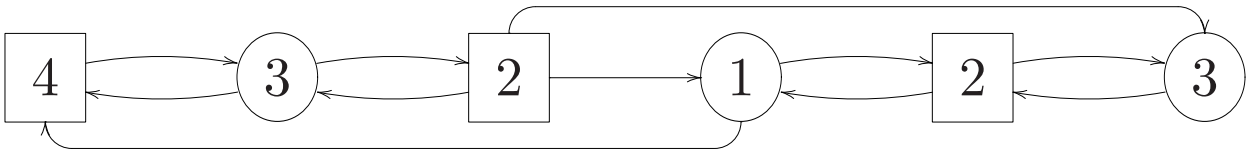
Kousha Etessami

What are Parity Games?

Parity games are a class of h.o. win-lose games on a graph:

$$G = (V_0, V_1, E, \lambda)$$

- **players**: P_0 (Even) and P_1 (Odd)
- **game graph**: (V, E), where $V = V_0 \cup V_1$
- **priority function**: λ: $V \to \mathbb{N}$
  value $\lambda(v)$ is called a *priority* of vertex v
- Notation: $\bigcirc \in V_0, \Box \in V_1$



What is the *winning condition*?

- play: π: $v_1 v_2 \ldots$
- $\lambda(\pi) = \lambda(v_1)\lambda(v_2)\ldots$
- The **top priority** of a play:
  $\mathrm{Top}(\pi) = \max\{p | p$ appears infinitely often in $\lambda(\pi)\}$
- a play π is **winning for player P_0 if $\mathrm{Top}(\pi)$ is even, and for player P_1 if $\mathrm{Top}(\pi)$ is odd.**

In other words, player P_0 (P_1) wins play π if the highest priority appearing infinitely often in $\lambda(\pi)$ is even (odd).

Why are parity games interesting?

- *Every h.o. win-lose game (i.e., every Muller game) can be converted to an* "equivalent" *parity game.*

  This can be done by attaching some data called a *Latest Appearance Record* to the current state (keeping track of some things about the *history* of the play so far).

  However the translation will cause worst-case exponential blow-up in the size of the game graph.

- Several important problems in automated verification (model checking) boil down to solving a parity game. (μ*-calculus Model checking.*)

- Special complexity-theoretic status:
  deciding which player has a winning strategy is known to be in NP $\cap$ co-NP, but not known to be in polynomial time.

More on parity games

- infinite duration games on graphs
- win-lose games of perfect information
- history oblivious, but not finitistic.
- but still, as we shall see, **memorylessly determined**.

For a parity game graph G, we let G_v denote the corresponding parity game that starts in state v.

We want to decide which player has a winning strategy, in G_v. (And to compute such a winning strategy.)

These win-lose games are determined (i.e., one player or the other has a winning strategy), because they can easily be seen to be particular examples of *Borel games*.

Memoryless determinacy

Main Theorem Parity games are memorylessly determined.

This will give us a simple exponential-time algorithm to solve parity games: for every memoryless strategy σ of P_0, try it against every memoryless strategy τ of player 1, and see if it beats each such strategy.

Proof. There are several different proofs of this theorem. The following is based on [Ehrenfeucht, Mycielski'73].

Finite duration parity game (FPG), $\tilde{G}_v$ is defined and played as a normal PG, G_v, except that the play stops once some vertex w is revisited. The winner depends on the maximum priority p on the loop from w to w. If p is even, P_0 wins, otherwise P_1 wins.

(FPG determinacy) The FPG game $\tilde{G}_v$ is determined (i.e., one player or the other has a winning strategy).

Follows easily from the Zermelo's theorem: every finite zero-sum game of perfect information is determined.

Finite duration parity game with special node z (FPG$_z$): Given G_v, and another vertex z, the game $\tilde{G}_{v,z}$, with "special" vertex z, is just like the FPG game except that the first time vertex z is visited, the "history is erased".

The winner depends again on the maximum priority p on the first "loop" created, but now if z is visited then this "loop" might be arrived at "later", because we ignore the history before z was visited.

Clearly, game $\tilde{G}_{v,z}$ is also a finite zero-sum game, and thus determined, meaning one player or the other has a winning strategy.

Furthermore, if z belongs to P_i, then clearly any optimal strategy for P_i is memoryless from the vertex z, because the vertex is only encountered at most ONCE, and "history" is ignored if it is encountered.

Theorem *The PG game G_v, its FPG versions $\tilde{G}_v$, and its FPG$_z$ version $\tilde{G}_{v,z}$, all have the same "value" (i.e., the same player has a winning strategy), for all vertices v and z.*

Furthermore, in all these games both players have optimal memoryless strategies.

Proof of memoryless determinacy

First, suppose that a strategy $\tilde{\sigma}$ is winning for player P_0 in the game $\tilde{G}_v$. From this, we construct a strategy σ that is winning for P_0 in G_v as follows: In σ, play just like in $\tilde{\sigma}$ until a loop L_k is created. Then, erase the loop from the history, and continue playing according to $\tilde{\sigma}$ from the last visited state of the loop (but without the loop as part of history).

"Set aside the loop, L_k, let $k := k + 1$.

Since $\tilde{\sigma}$ is winning in $\tilde{G}_v$ it means that every created "loop" $L_1, L_2, \ldots$ will have top priority **even**. Therefore, for the entire infinite play in G_v the top priority occuring infinitely often is also **even**.
(Because, the top priority occuring infinitely often must have the top priority in infinitely many such loops L_k.)

There was nothing special about player P_0. We could have done the same arguement for player P_1.

Therefore, the same player has a winning strategy in G_v and in $\tilde{G}_v$.

Next (**Key Point**), we claim that the same player has a winning strategy in G_v and $\tilde{G}_{v,z}$, for any z.

The reasoning is entirely the same: let $\tilde{\sigma}$ be a winning strategy for player P_0 in $\tilde{G}_{v,z}$, then we construct from it a winning strategy σ for P_0 in G_v by erasing loops from the "history", except that we treat the first occurence of z as erasing *all* history (not just the loop history).

But note: P_i playing out of z has a optimal memoryless move out of z in $\tilde{G}_{v,z}$. In other words, we can rephrase $\tilde{G}_{v,z}$, as the new game G'_v where we fix the edge out of z to be an optimal memoryless edge.

We have thus created a new game, G'_v, with one fewer vertex for player P_i which has more than one outedge, and which has the same value as the game $\tilde{G}_{v,z}$ (and thus G_v), and moreover, such that an optimal strategy in G'_v yields an optimal strategy in G_v.

Now, by induction, we can find a game G', which has no vertex of P_i with more than one outedge, and the (optimal) memoryless strategy for P_i in G' yields an optimal memoryless strategy for P_i in G_v. $\square$

Solving 1-player parity games in P-time

Fact Let G be a 1-player parity game (no vertices V_1 belonging to P_1). Then P_0 has a winning strategy starting at a vertex v in G iff there is a cycle reachable from v in which the top priority is **even**.

Proof: easy: if such a cycle exists, P_0 can force it and win. If not, P_0 can't win with any memoryless strategy (any such strategy creates one cycle). □

Deciding who has a winning strategy in a 1-player parity game can be solved efficiently in P-time.

We need to check whether there is a cycle with top priority *even* reachable from v:

Repeat

1. If the current top priority p' labeling any node is **odd**, eliminate nodes labeled p' the graph.
2. otherwise (top priority p is **even**), check if $\exists$ node u labeled p reachable from start v, such that u is *on a cycle*.
   If so, HALT: OUTPUT *"P_0 has a winning strat."*.
   If not, eliminate nodes labeled p from the graph.

Until (no nodes remain)
HALT: *"P_0 does not have a winning strat."*.

Implications for complexity of 2-player Parity Games

Now, since we know that we can solve 1-player parity games efficiently, in polynomial time, then to solve a 2-player parity game we can use the following approach:

1. "Nondeterministically guess" a memoryless optimal strategy σ' for one of the two players (say, for player P_1).

2. Check if in the remaining 1-player parity game player P_0 has a winning stretegy. If not, then (since the game is determined), that means σ' is a winning stretegy for player P_1.

Of course, we could have done the same with the roles of P_0 and P_1 reversed.

But how do we "nondeterminstically guess" a memoryless strategy?? There are exponentially many (2^m) such strategies for player P_0 (exponential in the number m of nodes controlled by player P_0), even if we only have 2 choices at each node.

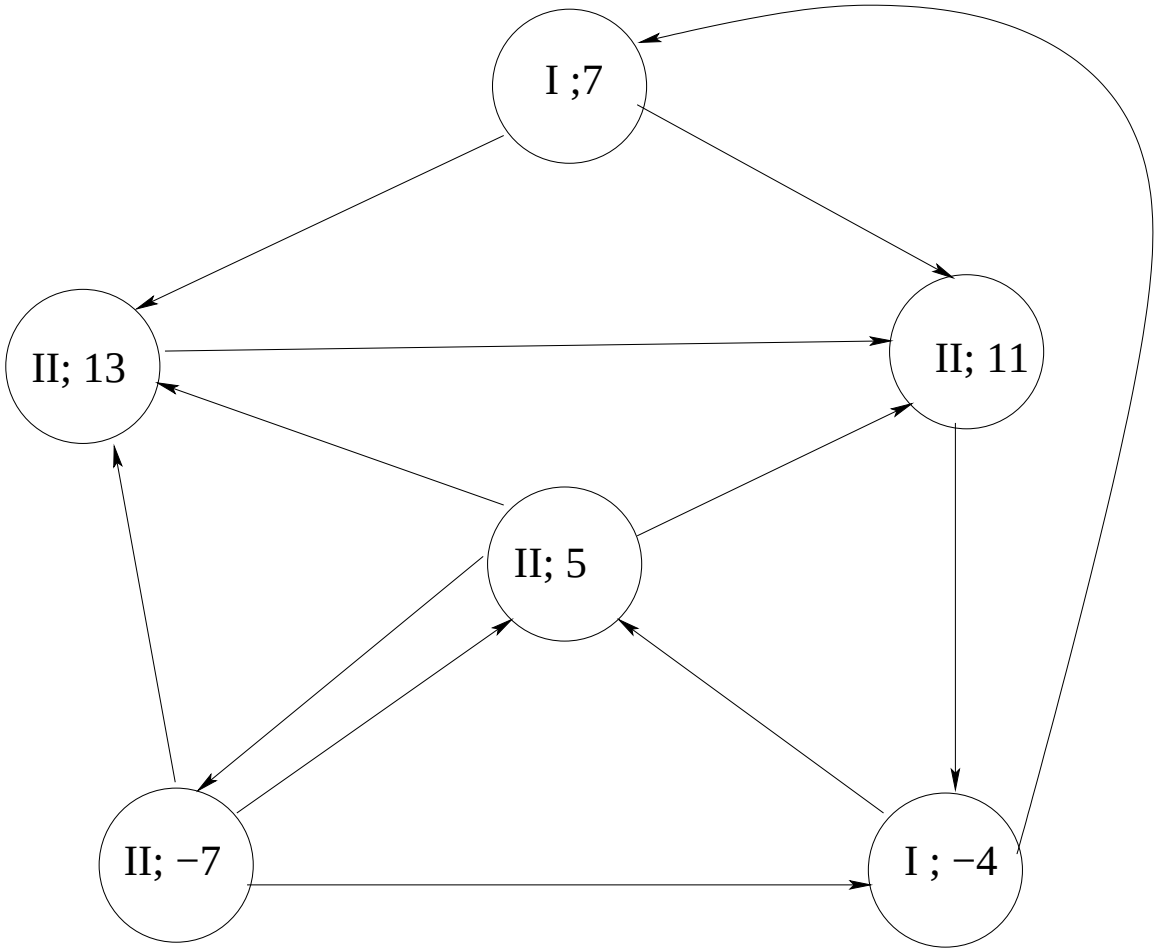
We can't do this deterministically in polynomial time, but we can do it in *Nondeterministic polynomial time*, and that is what the complexity class **NP**. (And **coNP** is the class of decision problems whose complement is in **NP**.)

Since we can do this for either player, this shows that the problem of deciding whether a player, say P_0, has a winning strategy in a given parity game is in the complexity class **NP** $\cap$ **coNP**.

(**N.B.** We will later learn that this problem is actually also reducible to computing a Nash equilibrium in a 2-player normal form game.)

Mean-Payoff Games

Every parity game can be reduced to a **mean payoff game**.



Zero-sum game.

Two players: I (**Max**) and II (**Min**)

Payoff (utility) function: $u : V \mapsto \mathbb{N}$ for player I.

Node $v \in V$ has payoff $u(v) \in \mathbb{N}$ for player I (Max).

Player I wants to maximize the following
limiting average (mean) payoff
in an infinite play $\pi = v_1 v_2 v_3 \ldots$:

$$\textit{Maximize:} \qquad \liminf_{n \to \infty} \frac{\sum_{i=1}^{n} u(v_i)}{n}$$

Thm: Mean-payoff games are *memoryless determined* (i.e., both players have deterministic memoryless strategies that achieve the value of the game.)

Proof: Basically the same proof as for parity games works.

The only modification needed is that in "Finite Mean-Payoff Games", we calculate the mean payoff of the finite play to be the *average payoff of those nodes on the single loop that was created when the game finished.*

In the finite mean-payoff game with a special node z, we treat z in exactly the same way as before: it "erases" history up to that point.

Everything then in the argument works: we can show that whatever mean-payoff either player, P_i, can force

in the finite game can also be forced in the infinite game. We do this by taking the average of the infinitely many "optimal loops" that will be created by the optimal strategy in the finite game when it is translated to the infinite mean-payoff game in a way analogous to the way we went from a strategy for finite parity games to infinite parity game.

In fact, this proof was originally devised for mean payoff games by Ehrenfeucht and Mycielski (1973). □

For both parity games and mean-payoff games, we do not know a worst-case efficient (polynomial time) algorithm. But we know that the relevant decision problems are in $\mathbf{NP} \cap \mathbf{coNP}$.

(1-player Mean-payoff games can be solved in polynomial time by an algorithm due to Karp (1978) for finding the *minimum mean-weight cycle* value in a directed graph.)