Polyhedral value iteration for discounted games and energy games

Alexander Kozachinskiy*

Department of Computer Science, University of Warwick, Coventry, UK

July 20, 2020

Abstract

We present a deterministic algorithm solving discounted games with $n$ nodes in strongly $n^{O(1)} \cdot (2 + \sqrt{2})^n$-time. For a special case of bipartite discounted games our algorithm runs in $n^{O(1)} \cdot 2^n$-time. Prior to our work no deterministic algorithm running in time $2^{o(n \log n)}$ regardless of the discount factor was known.

We call our approach polyhedral value iteration. We rely on a well-known fact that the values of a discounted game can be found from the so-called optimality equations. In the algorithm we consider a polyhedron obtained by relaxing optimality equations. We iterate the points on the border of this polyhedron by moving each time along a carefully chosen shift as far as possible. This continues until the current point satisfies optimality equations.

Our approach is heavily inspired by a recent algorithm of Dorfman et al. (ICALP 2019) for energy games. For completeness, we present their algorithm in terms of polyhedral value iteration. Our exposition, unlike the original algorithm, does not require edge weights to be integers and works for arbitrary real weights.

1 Introduction

We study discounted games, mean payoff games and energy games. All these three kinds of games are played on finite weighted directed graphs between two players called Max and Min. Players shift a pebble along the edges of a graph. Nodes of the graph are partitioned into two subsets, one where Max controls the pebble and the other where Min controls the pebble. One should also indicate in advance a starting node (a node where the pebble is located initially). By making infinitely many moves players

*Alexander.Kozachinskiy@warwick.ac.uk. Supported by the EPSRC grant EP/P020992/1 (Solving Parity Games in Theory and Practice).
give rise to an infinite sequence of edges \( e_1, e_2, e_3, \ldots \) of the graph (here \( e_i \) is the \( i \)th edge passed by the pebble). The outcome of the game is a real number determined by a sequence \( w_1, w_2, w_3, \ldots \), where \( w_i \) is the weight of the edge \( e_i \). We assume that outcome serves as the amount of fine paid by player Min to player Max. In other words, the goal of Max is to maximize the outcome and the goal of Min is to minimize it.

The outcome is computed differently in discounted, mean payoff and energy games.

- **the outcome of a discounted game** is
  \[
  \sum_{i=1}^{\infty} \lambda^{i-1} w_i,
  \]
  where \( \lambda \in (0, 1) \) is a fixed in advance real number called *discount factor*.

- **the outcome of a mean payoff game** is
  \[
  \limsup_{n \to \infty} \frac{w_1 + \ldots + w_n}{n}.
  \]

- **the outcome of an energy game** is
  \[
  \begin{cases} 
  1 & \text{the sequence } (w_1 + w_2 + \ldots + w_n), n \in \mathbb{N} \text{ is bounded from below}, \\
  0 & \text{otherwise},
  \end{cases}
  \]
  (we interpret outcome 1 as victory of Max and outcome 0 as victory of Min).

In all these three games every starting node has value, i.e., a real number \( \alpha \) such that (a) there is a Max’s strategy \( \sigma \) guaranteeing that the outcome is at least \( \alpha \) and (b) there is a Min’s strategy \( \tau \) guaranteeing that the outcome is at most \( \alpha \). Moreover \([24, 7, 5]\) we can always choose \( \sigma \) and \( \tau \) to be positional and independent of the starting node. Positionality means that strategy never makes two different moves in the same node. A property of having such \( \sigma \) and \( \tau \) is often called *positional determinacy*.

We study algorithmic problems that arise from these games. Namely, the *value problem* is a problem of finding values of a given game. The *decision problem* is a problem of comparing the value of a node with a given threshold. Another fundamental problem is to find positional strategies establishing the value of a game.

**Motivation.** Positionally determined games are of great interest in design of algorithms and computational complexity. Specifically, these games serve as a source of problems that are in \( \text{NP} \cap \text{coNP} \) but not known to be in \( \text{P} \).

Below we survey algorithms for discounted, mean payoff and energy games (including our contribution). Mean payoff and discounted games are also studied in context of dynamic systems \([9]\). Positionally determined games in general have a broad impact on formal languages and automata theory \([2]\).
Value problem vs. decision problem. The value problem, as a more general one, is at least as hard as the decision problem. On the other hand, the values in discounted and mean payoff games can be obtained from a play of two positional strategies. Hence, the bit-length of values is polynomial in bit-length of weights of edges and (in case of discounted games) bit-length of discount factor. This makes the value problem polynomial-time reducible to the decision problem via binary search. For energy games there is no difference between these two problems at all.

On the other hand, for discounted and mean payoff games the value problem may turn out to be harder for strongly polynomial algorithms. Indeed, in the reduction given above one manipulates directly with binary representations of weights (to identify a range containing values). This is prohibited for strongly polynomial algorithms.

Reductions, structural complexity. It is known that Max wins in an energy game if and only if the value of the corresponding mean payoff game is non-negative [3]. Hence, energy games are equivalent to decision problem for mean-payoff with threshold 0. Any other threshold $\alpha$ is reducible to threshold 0 by adding $-\alpha$ to all the weights. So energy games and mean payoff games are polynomial-time equivalent.

Decision problem for discounted games lies in UP $\cap$ coUP [15]. In turn, mean payoff games are polynomial-time reducible to discounted games [25]. Hence, the same UP $\cap$ coUP upper bound applies to mean payoff and energy games. None of these problems is known to lie in P.

Algorithms for discounted games. There are two classical approaches to discounted games. In value iteration approach, going back to Shapley [24], one manipulates with a real vector indexed by the nodes of the graph. The vector of values of a discounted game is known to be a fixed point of an explicit contracting operator. By applying this operator repeatedly to an arbitrary initial vector one obtains a sequence converging to the vector of values. Using this, Littman [17] gave a deterministic $O\left(\frac{n^{O(1)}}{1-\lambda} \log \left(\frac{1}{1-\lambda}\right)\right)$-time algorithm solving the value problem for discounted games. Here $n$ is the number of nodes, $\lambda$ is the discount factor and $L$ is the bit-length of input. This gives a polynomial time algorithm for $\lambda = 1 - \Omega(1)$.

Strategy iteration approach, going back to Howard [14] (see also [23]), can be seen as a sophisticated way of iterating positional strategies of players. Hansen et al. [13] showed that strategy iteration solves the value problem for discounted games in deterministic strongly $O\left(\frac{n^{O(1)}}{1-\lambda} \log_2 \left(\frac{1}{1-\lambda}\right)\right)$-time. Unlike Littman’s algorithm, for $\lambda = 1 - \Omega(1)$ this algorithm is strongly polynomial.

More recently, interior point methods we applied to discounted games [12]. As of now, however, these methods do not outperform the algorithm of Hansen et al.

For all these algorithms the running time depends on $\lambda$ (exponentially in the bit-length of $\lambda$). As far as we know, no deterministic algorithm with running time $2^{o(n \log n)}$ regardless of the value of $\lambda$ was known. One can get $2^{O(n \log n)}$ time by simply trying all possible positional strategies of one of the players. Our main result pushes this bound down to $2^{O(n)}$. More precisely, we show the following
**Theorem 1.** The values of a discounted game on a graph with \( n \) nodes can be found in deterministic strongly \( n^{O(1)} \cdot (2 + \sqrt{2})^n \)-time.

We also obtain a better bound for a special case of discounted games, namely for **bipartite** discounted games. We call a discounted game bipartite if in the underlying graph each edge is either an edge from a Max’s node to a Min’s node or an edge from a Min’s node to a Max’s node. In other words, in a bipartite discounted game players can only make moves alternatively.

**Theorem 2.** The values of a bipartite discounted game on a graph with \( n \) nodes can be found in deterministic strongly \( n^{O(1)} \cdot 2^n \)-time.

Our algorithm is the fastest known deterministic algorithm for discounted games when \( \lambda \geq 1 - (2 + \sqrt{2} + \Omega(1))^{-n} \). For bipartite discounted games it is the fastest one for \( \lambda \geq 1 - (2 + \Omega(1))^{-n} \). For smaller discounts the algorithm Hansen et al. outperforms ours. One should also mention that their algorithm is applicable to more general stochastic discounted games, while our algorithm is not.

In addition, it is known that randomized algorithms can solve discounted games faster, namely, in time \( 2^{O(\sqrt{n \log n})} \) [18, 11, 1]. These algorithms are based on formulating discounted games as an LP-type problem [19].

**Algorithms for mean payoff and energy games.** For mean payoff and energy games it is usually assumed that weights of edges are integers, and running time often involves a parameter \( W \), the largest absolute weight. In case of rational weights one can simply multiply them by a common denominator.

Zwick and Paterson [25] gave an algorithm solving the value problem for mean payoff games in pseudopolynomial time, namely, in time \( O(n^{O(1)} \cdot W) \) (see also [21]). Brim et al. [4] improved the polynomial factor before \( W \). In turn, Fijalkow et al. [8] slightly improved the dependence on \( W \) (from \( W \) to \( W^{1-1/n} \)).

There are algorithms with running time depending on \( W \) much better (at the cost that they are exponential in \( n \)). Lifshits and Pavlov [16] gave \( O(n^{O(1)} \cdot 2^n) \)-time algorithm for energy games (here the running time does not depend at all on \( W \)). Recently, Dorfman et al. [6] pushed \( 2^n \) down to \( 2^{n/2} \) by giving a \( O(n^{O(1)} \cdot 2^{n/2} \log W) \)-time algorithm for energy games. They also claim (without proof) that \( \log W \) factor can be removed. At the cost of an extra \( \log W \) factor these algorithms can be lifted to the value problem for mean payoff games.

All these algorithms are deterministic. As for randomized algorithms, the state-of-the-art is \( 2^{O(\sqrt{n \log n})} \)-time, the same as for discounted games.

We show that:

**Theorem 3.** For of an energy game on \( n \) nodes one can find all the nodes where Max wins in deterministic strongly \( n^{O(1)} 2^{n/2} \)-time.

This certifies that for the algorithm of Dorfman et al. \( \log W \) factor can be removed. More importantly, unlike the algorithm of Dorfman et al., our algorithm is strongly
\( n^{O(1)}2^{n/2}\)-time. I.e., our algorithm can be performed for arbitrary real weights (assuming basic arithmetic operations with them are carried out by an oracle).

The main reason we provide the proof of Theorem 3 is for the sake of exposition. Our result for discounted games is highly inspired by the Dorfman et al. algorithm. So we find it instructive to give Theorem 3 along with Theorem 1. We also believe that our exposition is more transparent for the reasons discussed below.

1.1 Our technique

Arguably, our approach arises more naturally for discounted games, yet it roots in the algorithm of Dorfman et al. for mean payoff games.

For discounted games we iterate a real vector \( x \) with coordinates indexed by the nodes of the graph, until \( x \) coincides with the vector of values. Thus, our approach can also be called value iteration. However, it differs significantly from the classical value iteration, and we call it polyhedral value iteration.

We rely on a well known fact that the vector of values is a unique solution to so-called optimality equations. Optimality equations is a set of conditions that can be naturally split into two parts. The first part is just a system of linear inequalities over \( x \), where each node has some subset of inequalities associated specifically with this node. They express the fact that the players can not improve the value in a node. The second part states that among inequalities associated with a node there is one turning into equality. This part represents the fact that values can be attained.

By throwing away the second part we obtain a polyhedron containing the vector of values. We call this polyhedron optimality polyhedron. Of course, besides the vector of values there are some other points too.

We initialize \( x \) by finding any point belonging to optimality polyhedron. There is little chance that \( x \) will satisfy optimality equations. So until it does, we do the following. We compute a shift directed from \( x \) to the interior of the optimality polyhedron. We move \( x \) along this shift as far as possible, until the border of optimality polyhedron is reached. This point on the border will be the new value of \( x \).

We choose a shift in a very specific way. We consider an auxiliary discrete game which we call discounted normal play game. The graph of the game depends on what inequalities of the optimality polyhedron are tight on \( x \). The values of this game determine a shift for \( x \). The rules of the game guaranty that such shift does not violate tight inequalities. Hence our shift does not immediately lead us outside the optimality polyhedron.

It turns out that this process converges to the vector of values. Moreover, it does in \( O(n(2 + \sqrt{2})^n) \) steps. The complexity analysis is split into two independent parts. First, we indicate some properties of how the underlying discounted normal play games are changing from one point to another in the algorithm. These leads to a definition of an abstract process of iterating discounted normal play games according to certain rules. In the second part of the argument we care only about this abstract process (called below DNP games iteration) and forget about the context of discounted games. We show that
DNP games iteration can last only $O(n(2 + \sqrt{2})^n)$ steps.

It turns out that in essentially the same language one can present the algorithm of Dorfman et al. Now we search not the solution to optimality equations but a vector of potentials certifying that one of the players wins in certain nodes. Dorfman et al. build upon a potential lifting algorithm of Brim et al. [4]. Dorfman et al. notice that in the algorithm of Brim et al. a lot of consecutive iterations may turn out to be lifting the same set of nodes. Instead, Dorfman et al. perform all these iterations at once, accelerating the algorithm of Brim et al. We notice that this can be seen as one step of polyhedral value iteration, but now for mean payoff games.

Polyhedron, inside which it all happens, is a limit of optimality polyhedrons as $\lambda \to 1$. This resembles a well-known representation of mean payoff games as a limit of discounted games, see, e.g., [22].

Again, the complexity analysis is carried out by considering DNP games iteration. In case of mean payoff games one can impose stronger restrictions on this abstract process, and this leads to a better bound.

DNP games iteration is implicit in the complexity analysis of Dorfman et al. We believe that "abstractization" makes their argument more transparent. It also might lead to some other applications besides discounted games.

## 2 Preliminaries

### 2.1 Discounted games

To specify a discounted game $G$ one has to specify

- a finite directed graph $G = (V, E)$ in which every node has at least one out-going edge, i.e., in which every node is not a sink;

- a partition of the set of nodes $V$ into two disjoint subsets $V_{\text{Max}}$ and $V_{\text{Min}}$;

- a weight function $w: E \to \mathbb{R}$;

- a real number $\lambda \in (0, 1)$ called the discount factor.

Discounted games are played between two players called Max and Min. There is a pebble which in each moment of time is located in one of the nodes of $G$. First, we have to specify a node $s \in V$ where the pebble is located initially. After that, at each move of the game the pebble is shifted along some edge of $G$ by one of the players. Namely, if currently the pebble is in a node $a \in V_{\text{Max}}$, then player Max has to move the pebble to some node $b \in V$ satisfying $(a, b) \in E$. Similarly, if currently the pebble is in a node $c \in V_{\text{Min}}$, then player Min has to move the pebble to some node $d \in V$ satisfying $(c, d) \in E$. Since in $G$ every node has at least one out-going edge, it is always possible to make a move.

By making infinitely many moves according to the rules above players obtain an infinite path of the graph $G$. If $e_1, e_2, e_3, \ldots$ are edges of this path (in the order they are
visited), then the outcome of the game $G$ is determined by the corresponding sequence of weights:

$$w_1 = w(e_1), w_2 = w(e_2), w_3 = w(e_3), \ldots$$

Namely, player Min pays to player Max a fine of size

$$\sum_{i=1}^{\infty} \lambda^{i-1} w_i.$$  \hspace{1cm} (1)

In other words, the goal of Max is to maximize (1) and the goal of Min is to minimize (1).

For any discounted game $G$ and for any starting node $s$ there exists a real number $x^*_s$, called the value of $G$ in the node $s$, such that:

- there is a Max's strategy guarantying that (1) is at least $x^*_s$;
- there is a Min's strategy guarantying that (1) is at most $x^*_s$.

Moreover, the values of $G$ can be found from the following system of equations called optimality equations:

$$x_a = \max_{e=(a,b) \in E} w(e) + \lambda x_b, \quad a \in V_{\text{Max}},$$  \hspace{1cm} (2)

$$x_a = \min_{e=(a,b) \in E} w(e) + \lambda x_b, \quad a \in V_{\text{Min}},$$  \hspace{1cm} (3)

where the system is over a real vector $x$ with coordinates indexed by the nodes of the graph. More specifically, (a) there exists exactly one solution $x^*$ to (2–3) and (b) for any node $s$ the value of $G$ in $s$ coincides with $x^*_s$.

This characterization of the values of discounted games goes back to Shapley [24]. Let us sketch Shapley’s argument for reader’s convenience. The fact that (2–3) has exactly one solution follows from Banach fixed point theorem. Observe that the set of solutions to (2–3) coincides with the set of fixed points of the following mapping:

$$\Delta: \mathbb{R}^V \rightarrow \mathbb{R}^V, \quad \Delta(x)_a = \begin{cases} \max_{e=(a,b) \in E} w(e) + \lambda x_b & a \in V_{\text{Max}}, \\ \min_{e=(a,b) \in E} w(e) + \lambda x_b & a \in V_{\text{Min}}. \end{cases}$$

It remains to notice that $\Delta$ is $\lambda$-contracting with respect to $\| \cdot \|_{\infty}$-norm.

Now, let $x^*$ be the solution to (2–3). We have to come up with a Max’s strategy $\sigma$ and a Min’s strategy $\tau$ proving that the value in the node $s$ exists and coincides with $x^*_s$. Let $\sigma$ be a strategy that from a node $a \in V_{\text{Max}}$ moves along an edge on which the maximum in (2) is attained. Similarly, let $\tau$ be a strategy that from a node $a \in V_{\text{Min}}$ moves along an edge on which the minimum in (3) is attained. It is not hard to verify that

- if the game starts in $s$ and Max follows $\sigma$, then (1) is at least $x^*_s$;
if the game starts in s and Min follows \( \tau \), then (1) is at most \( x^*_s \).

Remarkably, strategies \( \sigma \) and \( \tau \) do not depend on \( s \). Moreover, strategies \( \sigma \) and \( \tau \) are positional, i.e., the moves they make depend only on a current node and not on a path to this node. Thus, discounted games belong to a class of *positionally determined* games [10].

In this paper we are interested in an algorithmic problem of finding for a given discounted game \( G \) and for every \( s \in V \) the value of \( G \) in \( s \). By throwing away the context of discounted games one can simply say that we are interested in finding the solution to (2–3).

### 2.2 Energy games

Energy games [3, 5] are also played between two players called Max and Min. They have the same underlying mechanics as discounted games. Namely, the game takes place on a directed graph \( G = (V, E) \) (with no sinks) equipped with a partition of \( V \) into sets \( V_{\text{Max}} \) and \( V_{\text{Min}} \) and with a weight function \( w: E \to \mathbb{R} \). In the same way players produce an infinite sequence \( w_1, w_2, w_3, \ldots \) of weights of edges they visit. Now there is no discount factor and no fine paid by Min to Max. Instead, depending on the sequence \( w_1, w_2, w_3, \ldots \), either Max or Min wins. More precisely, player Max wins if the sequence of partial sums \( w_1 + w_2 + \ldots + w_n, n \in \mathbb{N} \) is bounded from below. Player Min wins otherwise.

Energy games are also positionally determined. More precisely, there is always a Max’s positional strategy \( \sigma \) and a Min’s positional strategy \( \tau \) such that for every starting node \( s \) either \( \sigma \) is a Max’s winning strategy or \( \tau \) is a Min’s winning strategy. This follows from positional determinacy of more general mean payoff games [7] and requires more elaborate argument than for discounted games.

It is instructive to provide a characterization of positional winning strategies in energy games in terms of cycles. First, by the weight of a cycle we mean the sum of weights of its edges. We call a cycle positive if its weight is positive. In the same way we define negative cycles, zero cycles and so on. Now, for a Max’s positional strategy \( \sigma \) let \( G^\sigma \) be a graph obtained from \( G \) by removing edges that start in \( V_{\text{Max}} \) and are not consistent with strategy \( \sigma \). I.e., in \( G^\sigma \) each node from \( V_{\text{Max}} \) has exactly one out-going edge, namely one used by \( \sigma \) in this node. It is easy to see that \( \sigma \) is winning for Max in energy game with starting node \( s \) if and only if in the graph \( G^\sigma \) only non-negative cycles are reachable from \( s \).

Similarly, for a Min’s positional strategy \( \tau \) one can define the graph \( G_\tau \), where only edges used by \( \tau \) are left for nodes in \( V_{\text{Min}} \). Then a strategy \( \tau \) is winning for Max in energy game starting in a node \( s \) if and only if only negative cycles are reachable from \( s \) in \( G_\tau \).

In this notation positional determinacy means that there is always a positional Max’s strategy \( \sigma \) and a positional Min’s strategy \( \tau \) such that for every node \( s \) either only non-negative cycles are reachable from \( s \) in \( G^\sigma \) or only negative cycles are reachable from \( s \) in \( G_\tau \).
We consider an algorithmic problem of finding all the nodes where Max wins (equiva-
ently, all the nodes where Min wins).

2.3 Bipartite graphs and games

In the paper we use term “bipartite” for directed graphs \( G = (V, E) \) equipped with a
partition of \( V \) into sets \( V_{\text{Max}} \) and \( V_{\text{Min}} \). Namely, we call a directed graph \( G = (V, E) \)
bipartite if \( E \subseteq V_{\text{Max}} \times V_{\text{Min}} \cup V_{\text{Min}} \times V_{\text{Max}} \). Next, by bipartite discounted game or
bipartite energy game we mean a game played on a bipartite graph.

3 \( n^{O(1)} \cdot (2 + \sqrt{2})^n \)-time algorithm for discounted games

In this section we give an algorithm establishing Theorem 1 and 2.

We consider a discounted game on a graph \( G = (V, E) \) with a weight function \( w: E \to \mathbb{R} \) and with a partition of \( V \) between the players given by the sets \( V_{\text{Max}} \) and \( V_{\text{Min}} \). We assume that \( G \) has \( n \) nodes and \( m \) edges.

In Subsection 3.1 we define auxiliary games that we call discounted normal play
games. We use these games both in the formulation of the algorithm and in the com-
plexity analysis. In Subsection 3.2 we define so-called optimality polyhedron by relaxing
optimality equations \( (2-3) \).

The algorithm is given in Subsection 3.3. In the algorithm we iterate the points of
the optimality polyhedron in search of the solution to \( (2-3) \). First we initialize by finding
any point belonging to the optimality polyhedron. Then for a current point we define
a shift which does not immediately lead us outside the optimality polyhedron. In the
definition of the shift we use discounted normal play games. To obtain the next point
we move as far as possible along the shift until we reach the border. We do so until the
current point satisfies \( (2-3) \). Along the way we also take some measures to prevent the
bit-length of the current point of growing super-polynomially.

This process always terminates and, in fact, can take only \( O(n(2 + \sqrt{2})^n) \) iterations.
Moreover, for bipartite discounted games it can take only \( O(2^n) \) steps. A proof of it is
defferred to Section 4.

3.1 Discounted normal play games.

These games will always be played on directed graphs with the same set of nodes as \( G \).
Given such a graph \( G' = (V, E') \), we equip it with the same partition of \( V \) into \( V_{\text{Max}} \) and
\( V_{\text{Min}} \) as in \( G \). There may be sinks in \( G' \).

Two players called Max and Min move a pebble along the edges of \( G' \). Player Max
controls the pebble in the nodes from \( V_{\text{Max}} \) and player Min controls the pebble in the
nodes from \( V_{\text{Min}} \). If the pebble reaches a sink of \( G' \) after \( s \) moves, then the player who can
not make a move pays fine of size \( \lambda^s \) to his opponent. Here \( \lambda \) is the discount factor from
our discounted game. If the pebble never reaches a sink, i.e., if the play lasts infinitely
long, then players pay each other nothing.
By the outcome of the play we mean the income of player Max. Thus, the outcome is

- positive, if the play ends in a sink from $V_{\text{Min}}$;
- zero, if the play lasts infinitely long;
- negative, if the play ends in a sink from $V_{\text{Max}}$.

It is not hard to see that in this game players have optimal positional strategies. Moreover, if $\delta(v)$ is the value of this game in the node $v$, then

$$
\delta(s) = -1, \text{ if } s \text{ is a sink from } V_{\text{Max}},
$$

$$
\delta(s) = 1, \text{ if } s \text{ is a sink from } V_{\text{Min}},
$$

$$
\delta(a) = \lambda \cdot \max_{(a,b) \in E'} \delta(b), \text{ if } a \in V_{\text{Max}} \text{ and } a \text{ is not a sink},
$$

$$
\delta(a) = \lambda \cdot \min_{(a,b) \in E'} \delta(b), \text{ if } a \in V_{\text{Min}} \text{ and } a \text{ is not a sink}.
$$

We omit proofs of these facts as below we only require the following

**Proposition 4.** For any $G = (V, E')$ there exists exactly one solution to (4–7), which can be found in strongly polynomial time.

Before proving Proposition 4 let us note that for graphs with $n$ nodes any solution $\delta$ to (5–6) satisfies $\delta(v) \in \{1, \lambda, \ldots, \lambda^{n-1}, 0, -\lambda^{n-1}, \ldots, -1\}$. Indeed, if $a$ is not a sink, then by (6–7) the node $a$ has an out-going edge leading to a node with $\delta(b) = \delta(a)/\lambda$. By following these edges we either reach a sink after at most $n - 1$ steps (and then $\delta(a) = \pm \lambda^i$ for some $i \in \{0, 1, \ldots, n - 1\}$) or we go to a loop. For all the nodes on a loop of length $l \geq 1$ we have $\delta(b) = \lambda^l \delta(b)$ which means that $\delta(b) = 0$ everywhere on the loop (recall that $\lambda \in (0, 1)$). Thus, if we reach such a loop from $a$, we also have $\delta(a) = 0$.

From this it is also clear that $\delta(v) = 1$ if and only if $v \in V_{\text{Min}}$ and $v$ is a sink of $G'$. Similarly, $\delta(v) = -1$ if and only if $v \in V_{\text{Max}}$ and $v$ is a sink of $G'$.

**Proof of Proposition 4.** To show the existence of a solution and its uniqueness we employ Banach fixed point theorem. Let $\Delta$ be the set of all vectors $f \in \mathbb{R}^V$, satisfying

$$
f(s) = 1 \text{ for all sinks } s \in V_{\text{Min}}, \quad f(t) = -1 \text{ for all sinks } t \in V_{\text{Max}}.
$$

Define the following mapping $\rho: \Delta \to \Delta$:

$$
\rho(f)(a) = \begin{cases} 
-1 & a \text{ is a sink from } V_{\text{Max}}, \\
1 & a \text{ is a sink from } V_{\text{Min}}, \\
\lambda \cdot \max_{(a,b) \in E} f(b) & a \in V_{\text{Max}} \text{ and } a \text{ is not a sink}, \\
\lambda \cdot \min_{(a,b) \in E} f(b) & a \in V_{\text{Min}} \text{ and } a \text{ is not a sink}.
\end{cases}
$$
The set of solutions to (4–7) coincides with the set of \( \delta \in \Delta \) such that \( \rho(\delta) = \delta \). It remains to notice that \( \rho \) is \( \lambda \)-contracting with respect to \( \| \cdot \|_\infty \)-norm.

Now let us explain how to find the solution to (4–7) in strongly polynomial time. Let us first determine for every \( k \in \{0, 1, \ldots, n - 1\} \) the set \( V_k = \{ v \in V \mid \delta(v) = \lambda^k \} \). It is clear that \( V_0 \) coincides with the set of sinks of the graph \( G' \) which lie in \( V_{\text{Min}} \). Next, the set \( V_k \) can be determined in strongly polynomial time once \( V_0, V_1, \ldots, V_{k-1} \) are given. Indeed, by (6–7) the set \( V_k \) consists of

- all \( v \in V_{\text{Max}} \setminus V_{<k} \) that have an out-going edge leading to \( V_{<k} \);
- all \( v \in V_{\text{Min}} \setminus V_{<k} \) such that all edges starting at \( v \) lead to \( V_{<k} \).

Here \( V_{<k} = V_0 \cup V_1 \cup \ldots \cup V_{k-1} \). In this way we determine all the sets \( V_0, V_1, \ldots, V_{n-1} \). Similarly way one can determine all the nodes with \( \delta(v) < 0 \) and also the exact value of \( \delta \) in these nodes. All the remaining nodes satisfy \( \delta(v) = 0 \). \( \square \)

### 3.2 Optimality polyhedron

By the optimality polyhedron we mean the set of all \( x \in \mathbb{R}^V \), satisfying the following inequalities:

\[
\begin{align*}
    w(e) + \lambda x_b - x_a & \leq 0 \quad \text{for} \; (a, b) \in E, a \in V_{\text{Max}}, \\
    w(e) + \lambda x_b - x_a & \geq 0 \quad \text{for} \; (a, b) \in E, a \in V_{\text{Min}}.
\end{align*}
\]

We denote the optimality polyhedron by \( \text{OptPol} \). Note that the solution to optimality equations (2–3) belongs to \( \text{OptPol} \).

We call a vector \( \delta \in \mathbb{R}^V \) a valid shift for \( x \in \text{OptPol} \) if for all small enough \( \varepsilon > 0 \) the vector \( x + \varepsilon \delta \) belongs to \( \text{OptPol} \). To determine whether a shift \( \delta \) is valid for \( x \) it is enough to look at the edges which are tight for \( x \). Namely, we call an edge \( (a, b) \in E \) tight for \( x \in \text{OptPol} \) if \( w(e) + \lambda x_b - x_a = 0 \), i.e., if the corresponding inequality in (8–9) becomes an equality on \( x \). It is clear that \( \delta \in \mathbb{R} \) is valid for \( x \) if and only if

\[
\begin{align*}
    \lambda \delta(b) - \delta(a) & \leq 0 \quad \text{whenever} \; (a, b) \in E, a \in V_{\text{Max}} \; \text{and} \; (a, b) \; \text{is tight for} \; x, \\
    \lambda \delta(b) - \delta(a) & \geq 0 \quad \text{whenever} \; (a, b) \in E, a \in V_{\text{Min}} \; \text{and} \; (a, b) \; \text{is tight for} \; x.
\end{align*}
\]

Discounted normal play games can be used to produce for any \( x \in \text{OptPol} \) a valid shift for \( x \). Namely, let \( E_x \subseteq E \) be the set of edges that are tight for \( x \) and consider the graph \( G_x = (V, E_x) \). I.e., \( G_x \) is a subgraph of \( G \) containing only edges that are tight for \( x \). An important observation is that \( x \) is the solution to optimality equations (2–3) if and only if in \( G_x \) there are no sinks.

Define \( \delta_x \) to be the solution to (4–7) for \( G_x \). Note that \( \delta_x \) is a valid shift for \( x \) as (6–7) imply (10–11). Not also that as long \( x \) does not satisfy (2–3), i.e., as long as the graph \( G_x \) has sinks, the vector \( \delta_x \) is not zero.

Let us also define a procedure \( \text{RealizeGraph}(S) \) that we use in our algorithm to control the bit-length of the current point. The input to the procedure is a subset \( S \subseteq E \).
The output of $\text{RealizeGraph}(S)$ is a point $x \in \text{OptPol}$ satisfying $S \subseteq E_x$. If there is no such $x$, the output of $\text{RealizeGraph}(S)$ is “not found”. In other words, consider a polyhedron which can be obtained from (8–9) by turning inequalities corresponding to edges from $S$ into equalities. The output of $\text{RealizeGraph}(S)$ is a point of this polyhedron, if this polyhedron is not empty. In particular, $\text{RealizeGraph}(\emptyset)$ is simply a procedure of finding a point belonging to $\text{OptPol}$.

Note that each inequality in (8–9) contains exactly two variables. Hence (see [20]), the output of $\text{RealizeGraph}(S)$ can be computed in strongly polynomial time.

### 3.3 The algorithm

**Algorithm 1: $n^{O(1)} \cdot (2 + \sqrt{2})^n$-time algorithm for discounted games**

| Result: The solution to optimality equations (2–3) |
|---------|
| initialization: $x = \text{RealizeGraph}(\emptyset)$; |
| while $x$ does not satisfy (2–3) do |
| $\varepsilon_{\text{max}} \leftarrow$ the largest $\varepsilon \in (0, +\infty)$ s.t. $x + \varepsilon \delta_x \in \text{OptPol}$; |
| $x \leftarrow \text{RealizeGraph}(E_x + \varepsilon_{\text{max}} \delta_x)$; |
| end |
| output $x$; |

Some remarks:

- we can find $\delta_x$ in strongly polynomial time by Proposition 4;

- the value of $\varepsilon_{\text{max}}$ can be found as in the simplex-method. Indeed, $\varepsilon_{\text{max}}$ is the smallest $\varepsilon \in (0, +\infty)$ for which there exists an inequality in (8–9) which is tight for $x + \varepsilon \delta_x$, but not for $x$. Thus, to find $\varepsilon_{\text{max}}$ it is enough to solve at most $m$ linear one-variable equations and compute the minimum over positive solutions to these equations.

- in fact, $\varepsilon_{\text{max}} < +\infty$ throughout the algorithm, i.e, we can not move along $\delta_x$ forever. To show this, it is enough to indicate $\varepsilon > 0$ and an inequality in (8–9) which is tight for $x + \varepsilon \delta_x$ but not for $x$. First, since $x$ does not yet satisfy optimality equations (2–3), there exists a sink $s$ of the graph $G_x$. Assume that $s \in V_{\text{Max}}$, the argument in the case $s \in V_{\text{Min}}$ is similar. The graph $G$ is sinkless, so there exists an edge $e = (s, b) \in E$. The edge $(s, b)$ is not tight for $x$ (otherwise $s$ is not a sink of $G_x$). Hence $w(e) + \lambda x_b - x_s < 0$. The left-hand side of the same inequality for $x + \varepsilon \delta_x$ looks as follows:

$$w(e) + \lambda x_b - x_s + \varepsilon \cdot (\lambda \delta_x(b) - \delta_x(s)).$$

In turn, the node $s$ is a sink of $G_x$ from $V_{\text{Max}}$, hence $\delta_x(s) = -1 < \lambda \delta_x(b)$. I.e., the left-hand side of the inequality for the edge $(s, b)$ increases as $\varepsilon$ increases, so for some positive $\varepsilon$ it will become tight.
• One could consider a version of the Algorithm 1 where we do not use the procedure \textit{RealizeGraph} and simply set \( x \leftarrow x + \epsilon \max \delta x \). A problem with this version is that it is not clear why the bit-length of the coordinates of \( x \) \( a \) is polynomially bounded throughout the algorithm. In turn, if we use the procedure \textit{RealizeGraph}, this problem does not occur. Indeed, we maintain the property that \( x \) is an output of a strongly polynomial time algorithm on a polynomially bounded input.

4 Discounted games: complexity analysis

Let \( x_0, x_1, x_2, \ldots \) be a sequence of points from \textit{OptPol} that arise in the Algorithm 1. The argument consists of two parts:

• first, we show that the sequence of graph \( G_{x_0}, G_{x_1}, G_{x_2} \ldots \) can be obtained in an abstract process that we call discounted normal play games iteration (DNP games iteration for short), see Subsection 4.2;

• second, we show that any sequence of \( n \)-node graphs that can be obtained in DNP games iteration has length \( O(n(2 + \sqrt{2})^n) \), see Subsection 4.3.

This will establish Theorem 1. To show Theorem 2 note that if \( G \) is bipartite, then so are \( G_{x_0}, G_{x_1}, G_{x_2} \) and so on. Thus, it is enough to demonstrate that:

• any sequence of bipartite \( n \)-node graphs that can be obtained in DNP games iteration has length \( O(2^n) \), see Subsection 4.4.

First of all, we have to give a definition of DNP games iteration (Subsection 4.1).

4.1 Definition of DNP games iteration

Consider a directed graph \( H = (V, E_1) \) and let \( \delta_H \) be the solution to (4–7) for \( H \). We say that the edge \((a, b) \in E_1\) is \textit{optimal} for \( H \) if \( \delta_H(a) = \lambda \delta_H(b) \). Next, we say that the pair \((a, b) \in V \times V\) is \textit{improving} for \( H \) if one of the following two conditions holds:

• \( a \in V_{max} \) and \( \delta_H(a) < \lambda \delta_H(b) \);

• \( a \in V_{min} \) and \( \delta_H(a) > \lambda \delta_H(b) \).

Note that an improving pair of nodes can not be an edge of \( H \) because of (6–7).

Consider another directed graph \( K = (V, E_2) \) over the same set of nodes as \( H \). We say that \( K \) \textit{can be obtained from} \( H \) \textit{in one step of DNP games iteration} if \( E_2 \) contains all edges of \( H \) that are optimal for \( H \) and also at least one pair of nodes which is improving for \( H \). I.e., we can erase some non-optimal edges of \( H \), and then we can add some edges that are not in \( H \), in particular, we should add at least one improving pair.

Finally, we say that a sequence of graph \( H_0, H_1, \ldots, H_j \) \textit{can be obtained in DNP games iterations} if for all \( i \in \{0, 1, \ldots, j-1\} \) the graph \( H_{i+1} \) can be obtained from \( H_i \) in one step of DNP games iteration.
4.2 Why the sequence $G_{x_0}, G_{x_1}, G_{x_2}, \ldots$ can be obtained in DNP games iteration

Let $x$ and $x' = \text{RealizeGraph}(E_x + \varepsilon_{\max} \delta_x)$ be two consecutive points of OptPol in the algorithm. We have to show that the graph $G_{x'}$ can be obtained from $G_x$ in one step of DNP games iteration. By definition of the procedure RealizeGraph the graph $G_{x'}$ contains all edges of the graph $G_y$, where $y = x + \varepsilon_{\max} \delta_x$. Hence it is enough to show the following:

(a) all the edges of the graph $G_x$ that are optimal for $G_x$ are also in the graph $G_y$;
(b) there is an edge of the graph $G_y$ which is an improving pair for the graph $G_x$.

Proof of (a). Take any edge $(a, b)$ of the graph $G_x$ which is optimal for $G_x$. The left-hand side of (8-9) for the edge $(a, b)$ on the point $y = x + \varepsilon_{\max} \delta_x$ looks as follows:

$$w(e) + \lambda x_b - x_a + \varepsilon_{\max} \cdot (\lambda \delta_x(b) - \delta_x(a)).$$ (12)

The last term of (12) is 0 as $(a, b)$ is an optimal edge of $G_x$. Since $(a, b)$ is tight for $x$, it is also tight for $y$, i.e., it also belongs to $G_y$.

Proof of (b). In fact, any edge of the graph $G_y$ which is not in the graph $G_x$ is an improving pair for $G_x$. Assume $(a, b) \in E$ is an edge of $G_y$ but not of $G_x$. Hence $(a, b)$ is tight for $y$ but not for $x$. I.e., (12) is 0 for $(a, b)$, but

- $w(e) + \lambda x_b - x_a < 0$ if $a \in V_{\text{Max}}$;
- $w(e) + \lambda x_b - x_a > 0$ if $a \in V_{\text{Min}}$.

This means that $\lambda \delta_x(b) - \delta_x(a) > 0$ if $a \in V_{\text{Max}}$ and $\lambda \delta_x(b) - \delta_x(a) < 0$ if $a \in V_{\text{Min}}$. Therefore $(a, b)$ is an improving pair for $G_x$.

It only remains to note that there exists an edge of $G_y$ which is not an edge of $G_x$. Indeed, otherwise all inequalities that are tight for $y = x + \varepsilon_{\max} \delta_x$ were tight already for $x$. Then $\varepsilon_{\max}$ could be increased, contradiction.

4.3 $O(n(2 + \sqrt{2})^n)$ bound on the length of DNP games iteration

The argument has the following structure.

- Step 1. For every directed graph $H = (V, E_1)$ we define two vectors $f^H, g^H \in \mathbb{N}^{2n-1}$.
- Step 2. We define a linear ordering of vectors from $\mathbb{N}^{2n-1}$ called alternating lexicographic ordering.
- Step 3. We show that in each step of DNP games iteration (a) neither $f^H$ nor $g^H$ decrease and (b) either $f^H$ or $g^H$ increase (in the alternating lexicographic ordering).
Step 4. We bound the number of values \( f^H \) and \( g^H \) can take. By step 3 this bound (multiplied by 2) is also a bound on the length of DNP games iteration.

**Step 1.** The first coordinate of the vector \( f^H \) equals the number of nodes with \( \delta_H(a) = 1 \) (all such nodes are from \( V_{\text{Min}} \)). The other \( 2n - 2 \) coordinates are divided into \( n - 1 \) consecutive pairs. In the \( i \)th pair we first have the number of nodes from \( V_{\text{Max}} \) with \( \delta_H(a) = \lambda^i \), and then the number of nodes from \( V_{\text{Min}} \) with \( \delta_H(a) = \lambda^i \).

The vector \( g^H \) is defined similarly, with the roles of Max and Min and + and − reversed. The first coordinate of \( g^H \) equals the number of nodes with \( \delta_H(a) = -1 \) (all such nodes are from \( V_{\text{Max}} \)). The other \( 2n - 2 \) coordinates are divided into \( n - 1 \) consecutive pairs. In the \( i \)th pair we first have the number of nodes from \( V_{\text{Min}} \) with \( \delta_H(a) = -\lambda^i \), and then the number of nodes from \( V_{\text{Max}} \) with \( \delta_H(a) = -\lambda^i \).

**Step 2.** Alternating lexicographic ordering is a lexicographic order obtained from the standard ordering of integers in the even coordinates and from the reverse of the standard ordering of integers in the odd coordinates. For example,

\[
(3, 2, 3) < (2, 3, 2), \quad (2, 3, 1) > (2, 2, 7),
\]

in the alternating lexicographic order.

**Step 3.** This step relies on the following

**Lemma 5.** Assume that a graph \( K \) can be obtained from a graph \( H \) in one step of DNP games iteration. Then

(a) if for some \( i \in \{0, 1, \ldots, n - 1\} \) it holds that \( \{a \in V \mid \delta_H(a) = \lambda^i\} \neq \{a \in V \mid \delta_K(a) = \lambda^i\} \), then \( f^K \) is greater than \( f^H \) in the alternating lexicographic order.

(b) if for some \( i \in \{0, 1, \ldots, n - 1\} \) it holds that \( \{a \in V \mid \delta_H(a) = -\lambda^i\} \neq \{a \in V \mid \delta_K(a) = -\lambda^i\} \), then \( g^K \) is greater than \( g^H \) in the alternating lexicographic order.

Assume Lemma 5 is proved.

- Why neither \( f^H \) nor \( g^H \) can decrease? If \( f^K \) does not exceed \( f^H \) in the alternating lexicographic order, then \( \{a \in V \mid \delta_H(a) = \lambda^i\} \neq \{a \in V \mid \delta_K(a) = \lambda^i\} \) for every \( i \in \{0, 1, \ldots, n - 1\} \) by Lemma 5. On the other hand, \( f^H \) and \( f^K \) are determined by these sets, so \( f^H = f^K \). Similar argument works for \( g^H \) and \( g^K \) as well.

- Why either \( f^H \) or \( g^H \) increase? Assume that neither \( f^K \) is greater than \( f^H \) nor \( g^K \) is greater than \( g^H \) in alternating lexicographic order. By Lemma 5 we have for every \( i \in \{0, 1, \ldots, n - 1\} \) that \( \{a \in V \mid \delta_H(a) = \lambda^i\} = \{a \in V \mid \delta_K(a) = \lambda^i\} \) and \( \{a \in V \mid \delta_H(a) = -\lambda^i\} = \{a \in V \mid \delta_K(a) = -\lambda^i\} \). This means that functions \( \delta_H \) and \( \delta_K \) coincide. On the other hand, the graph \( K \) contains as an edge a pair of nodes which is improving for \( H \). Since \( \delta_K = \delta_H \), this means that this pair is also improving for \( K \). Hence this pair can not be an edge of the graph \( K \), contradiction.
We now proceed to a proof of Lemma 5. Let us stress that in the proof we do not use the fact that $K$ contains an improving pair for $H$. We only use the fact that $K$ contains all optimal edges of $H$.

**Proof of Lemma 5.** We only prove (a), the proof of (b) is similar. Let $j$ be the smallest element of $\{0, 1, \ldots, n - 1\}$ for which $\{a \in V \mid \delta_H(a) = \lambda^j\} \neq \{a \in V \mid \delta_K(a) = \lambda^j\}$. First consider the case $j = 0$. We claim that in this case the first coordinate of $f^K$ is smaller than the first coordinate of $f^H$. Indeed, $f^K_1$ is the number of sinks from $V_{\text{Min}}$ in the graph $H$. In turn, $f^K_1$ is the number of sinks from $V_{\text{Min}}$ in the graph $K$. On the other hand, there are sinks of $K$ that are also sinks of $H$. Indeed, nodes that are not sinks of $H$ have in $H$ an out-going optimal edge. All these edges are also in $K$. Hence $f^K_1 \leq f^H_1$. The equality is not possible because otherwise $\{a \in V \mid \delta_H(a) = 1\} \neq \{a \in V \mid \delta_K(a) = 1\}$, contradiction with the fact that $j = 0$.

Now assume that $j > 0$. Then the sets $\{v \in V \mid \delta_H(v) = \lambda^j\}$ and $\{v \in V \mid \delta_K(v) = \lambda^j\}$ are distinct. Hence there are two cases.

- **First case:** $\{v \in V_{\text{Max}} \mid \delta_H(v) = \lambda^j\} \neq \{v \in V_{\text{Max}} \mid \delta_H(v) = \lambda^j\}$.
- **Second case:** $\{v \in V_{\text{Max}} \mid \delta_H(v) = \lambda^j\} = \{v \in V_{\text{Max}} \mid \delta_H(v) = \lambda^j\}$ and $\{v \in V_{\text{Min}} \mid \delta_H(v) = \lambda^j\} \neq \{v \in V_{\text{Min}} \mid \delta_H(v) = \lambda^j\}$.

In both cases the first $1 + 2(j - 1)$ coordinates of $f^H$ and $f^K$ coincide, because $\{v \in V \mid \delta_H(v) = \lambda^j\} = \{v \in V \mid \delta_K(v) = \lambda^j\}$ for all $i < j$. Moreover, in the second case we also have $f^K_1 < f^H_1$. We claim that in the first case we have $f^K_{2j} < f^H_{2j}$ and in the second case we have $f^K_{2j+1} > f^H_{2j+1}$. The rest is devoted a proof of this claim as it clearly implies that $f^K$ exceeds $f^H$ in alternating lexicographic order.

**Proving $f^K_{2j} < f^H_{2j}$ in the first case.** Since the sets $\{v \in V_{\text{Max}} \mid \delta_H(v) = \lambda^j\}$ and $\{v \in V_{\text{Max}} \mid \delta_K(v) = \lambda^j\}$ are distinct, it is enough to show that $\{v \in V_{\text{Max}} \mid \delta_H(v) = \lambda^j\} \subseteq \{v \in V_{\text{Max}} \mid \delta_K(v) = \lambda^j\}$. For that we take any $a \in V_{\text{Max}}$ with $\delta_H(a) = \lambda^j$ and show that $\delta_K(a) = \lambda^j$. By (6–7) there is an edge $(a, b)$ of the graph $H$ with $\delta_H(b) = \lambda^{j-1}$. We also have that $\delta_K(b) = \lambda^{j-1}$, because $\{v \in V \mid \delta_H(v) = \lambda^{j-1}\} = \{v \in V \mid \delta_K(v) = \lambda^{j-1}\}$. On the other hand, since $\delta_H(a) = \lambda \delta_H(b)$, the edge $(a, b)$ is optimal for $H$, hence this edge is also in the graph $K$. So in the graph $K$ there is an edge from $a \in V_{\text{Max}}$ to a node $b$ with $\delta_K(b) = \lambda^{j-1}$. Hence by (6) we have $\delta_K(a) \geq \lambda^j$. It remains to show why it is impossible that $\delta_K(a) > \lambda^j$. Indeed, then $a \in \{v \in V \mid \delta_K(v) = \lambda^i\}$ for some $i < j$. On the other hand, the node $a$ is not in the set $\{v \in V \mid \delta_H(v) = \lambda^j\}$. Hence the sets $\{v \in V \mid \delta_H(v) = \lambda^j\}$ and $\{v \in V \mid \delta_K(v) = \lambda^j\}$ are distinct, contradiction with the minimality of $j$.

**Proving $f^K_{2j+1} > f^H_{2j+1}$ in the second case.** Since the sets $\{v \in V_{\text{Min}} \mid \delta_H(v) = \lambda^j\}$ and $\{v \in V_{\text{Min}} \mid \delta_K(v) = \lambda^j\}$ are distinct, it is enough to show that $\{v \in V_{\text{Min}} \mid \delta_K(v) = \lambda^j\} \subseteq \{v \in V_{\text{Min}} \mid \delta_H(v) = \lambda^j\}$. For that we take any $a \in V_{\text{Min}}$ with $\delta_K(a) = \lambda^j$ and show that $\delta_H(a) = \lambda^j$. It is clear that $\delta_H(a) \leq \lambda^j$, because otherwise for some $i < j$ we would have that the sets $\{v \in V \mid \delta_H(v) = \lambda^i\}$ and $\{v \in V \mid \delta_K(v) = \lambda^i\}$ are distinct ($a$ would belong to the first set and not to the second one). This would give us a contradiction with the minimality of $j$. Thus, it remains to show that $\delta_H(a) \geq \lambda^j$. Assume that this
is not the case, i.e., \( \delta_H(a) \leq \lambda_i^{j+1} \). Since \( a \in V_{\text{Min}} \), the node \( a \) is not a sink of \( H \) (this would mean that \( \delta_H(a) = 1 > \lambda_i^{j+1} \)). Hence by (7) there exists an edge \((a, b)\) in the graph \( H \) with \( \delta_H(b) = \delta_H(a)/\lambda \leq \lambda_i^j \). Then we also have that \( \delta_K(b) \leq \lambda_i^j \), because by minimality of \( j \) we have \( \{v \in V \mid \delta_H(a) > \lambda_i^{j-1}\} = \{v \in V \mid \delta_K(a) > \lambda_i^{j-1}\} \) and hence \( \{v \in V \mid \delta_H(a) \leq \lambda_i^j\} = \{v \in V \mid \delta_K(a) \leq \lambda_i^j\} \). But the edge \((a, b)\) is optimal for \( H \), so the edge \((a, b)\) is also in the graph \( K \). This means that in the graph \( K \) there is an edge from \( a \) to a node \( b \) with \( \delta_K(b) \leq \lambda_i^j \). Hence by (7) we have \( \delta_K(a) \leq \lambda_i^{j+1} \), contradiction.

\( \square \)

Step 4. Notice that \( f^H \) and \( g^H \) belong to the set of all vectors \( v \in \mathbb{N}^{2n-1} \) satisfying:

\[
\|v\|_1 \leq n, \quad (13)
\]

\[
v_1 = 0 \implies v_2 = v_3 = \ldots = v_{2n-1} = 0, \quad (14)
\]

\[
v_{2i} = v_{2i+1} = 0 \implies v_{2i+2} = v_{2i+3} = \ldots = v_{2n-1} = 0 \text{ for every } i \in \{1, \ldots, n-2\}. \quad (15)
\]

To see (13) note that in our case the \( l_1 \)-norm is just a sum of coordinates. By construction, the sum of coordinates of \( f^H \) is the number of nodes with \( \delta_H(a) > 0 \) and the sum of coordinates of \( g^H \) is the number of nodes with \( \delta_H(a) < 0 \). The fact that \( f^H \) satisfies (14–15) can be seen from the following observation: if \( \{a \in V : \delta_H(a) = \lambda_i^j\} = \emptyset \), then we also have \( \{a \in V : \delta_H(a) = \lambda_i^j\} = \emptyset \) for every \( j > i, j \in \{0, 1, \ldots, n-1\} \). Indeed, by (6–7) a node with \( \delta_H(a) = \lambda_i^j \) has an edge leading to a node with \( \delta_H(b) = \lambda_i^{j-1} \). By continuing in this way we would reach a node with \( \delta_H(a) = \lambda_i^i \), contradiction.

Thus, the desired upper bound on the length of DNP games iteration follows from the following technical lemma.

Lemma 6. The number of vectors \( v \in \mathbb{N}^{2n-1} \) satisfying (13–15) is \( O(n(2 + \sqrt{2})^n) \).

Proof. Let \( \mathcal{A} \) be the set of all vectors \( v \in \mathbb{N}^{2n-1} \) satisfying (13–15). For \( v \in \mathcal{A} \) let \( t(v) \) be the largest \( t \in \{1, 2, \ldots, n-1\} \) such that \( v_{2t} + v_{2t+1} > 0 \). If there is no such \( t \) at all (i.e., if \( v_2 = v_3 = \ldots = v_{2n-1} = 0 \)), then define \( t(v) = 0 \).

Let \( \mathcal{A}_t = \{v \in \mathcal{A} : t(v) = t\} \). We claim that \( |\mathcal{A}_t| \leq (2 + \sqrt{2})^n \) for any \( t \). As \( t(v) \) can take only \( O(n) \) values, the lemma follows.

The size of \( \mathcal{A}_0 \) is \( n \), so we may assume that \( t > 0 \). Take any \( \rho \in (0, 1) \). Observe that:

\[
\rho^n |\mathcal{A}_t| \leq \sum_{v \in \mathcal{A}_t} \rho^{\|v\|_1} = \sum_{v \in \mathcal{A}_t} \rho^{v_1} \cdot \rho^{v_2+v_3} \cdot \ldots \cdot \rho^{v_{2t}+v_{2t+1}}
\]

\[
\leq \left( \sum_{v_1=1}^{\infty} \rho^{v_1} \right) \left( \sum_{(v_2,v_3) \in \mathbb{N}^2 \setminus \{(0,0)\}} \rho^{v_2+v_3} \right) \ldots \left( \sum_{(v_{2t},v_{2t+1}) \in \mathbb{N}^2 \setminus \{(0,0)\}} \rho^{v_{2t}+v_{2t+1}} \right)
\]

\[
= \left( \sum_{a=1}^{\infty} \rho^{a} \right) \left( \sum_{(b,c) \in \mathbb{N}^2 \setminus \{(0,0)\}} \rho^{b+c} \right)^t.
\]

Indeed, the first inequality here holds because \( \|v\|_1 \leq n \) by (13) for \( v \in \mathcal{A} \). The second inequality holds because for \( v \in \mathcal{A} \) with \( t(v) = t \) we have \( v_1 > 0 \) by (14) and \( v_{2i}+v_{2i+1} > 0 \) for every \( i \in \{1, 2, \ldots, t\} \) by (15).
Next, notice that for $\rho = 1 - \frac{1}{\sqrt{2}}$ we have:

\[
\left( \sum_{a=1}^{\infty} \rho^a \right) \cdot \left( \sum_{(b,c)\in\mathbb{N}^2\setminus\{(0,0)\}} \rho^{b+c} \right)^t \leq 1
\]

Indeed,

\[
\sum_{a=1}^{\infty} \rho^a = \frac{\rho}{1-\rho} = \sqrt{2} - 1 < 1,
\]

\[
\sum_{(b,c)\in\mathbb{N}^2\setminus\{(0,0)\}} \rho^{b+c} = \frac{1}{(1-\rho)^2} - 1 = 1.
\]

Thus, we get $\rho^n|\mathcal{A}_t| \leq 1$. I.e., $|\mathcal{A}_t| \leq (1/\rho)^n = (2 + \sqrt{2})^n$, as required.

In fact, as shown in Appendix A, Lemma 6 is tight up to a polynomial factor.

### 4.4 $O(2^n)$ bound on the length of DNP games iteration for bipartite graphs

The proof differs only in the last step, where for bipartite graphs we obtain a better bound. In more detail, if $H$ is bipartite, then $f^H$ and $g^H$ in addition to (13–15) satisfy the following property:

\[
u_{2i} = 0 \text{ for even } i \in \{1, \ldots, n-1\}, \quad \nu_{2i+1} = 0 \text{ for odd } i \in \{1, \ldots, n-1\}.
\]

Indeed, for $f^H$ the condition (16) looks as follows:

\[
f^H_{2i} = |\{v \in V_{\text{Max}} \mid \delta_H(v) = \lambda^i\}| = 0 \text{ for even } i,
\]

\[
f^H_{2i+1} = |\{v \in V_{\text{Min}} \mid \delta_H(v) = \lambda^i\}| = 0 \text{ for odd } i.
\]

This holds because from a node $a$ with $\delta_H(a) = \lambda^i$ there a path of length $i$ to a node $s$ with $\delta_H(s) = 1$. If $\delta_H(s) = 1$, then $s \in V_{\text{Min}}$. Since $H$ is bipartite, this means that $a \in V_{\text{Min}}$ for even $i$ and $a \in V_{\text{Max}}$ for odd $i$. The argument for $g^H$ is the same.

So it is enough to show that the number of $v \in \mathbb{N}^{2n-1}$ satisfying (13–16) is $O(2^n)$. Let $t(v)$ be defined in the same way as in the proof of Lemma 6. I.e., $t(v)$ is the largest $t \in \{1, \ldots, n-1\}$ for which $v_{2i} + v_{2i+1} > 0$ (if there is no such $t$, we set $t(v) = 0$). Let us bound the number of $v \in \mathbb{N}^{2n-1}$ satisfying (13–16) and $\|v\|_1 = s, t(v) = t$.

For $t = 0$ the number of such $v$ is exactly 1. Assume now that $t > 0$. Then

\[
v_1 > 0, \quad v_{2i} > 0 \text{ and } v_{2i+1} = 0, \text{ for odd } i \in \{1, \ldots, t\}
\]

\[
v_{2i} = 0 \text{ and } v_{2i+1} > 0, \text{ for even } i \in \{1, \ldots, t\}
\]

\[
v_j = 0 \text{ for } j > 2t + 1,
\]

by definition of $t(v)$. 

18
Hence the number of $v \in \mathbb{N}^{2n-1}$ satisfying (13–16) and $\|v\|_1 = s, t(v) = t$ is equal to the number of the solutions to the following system:

$$x_1 + x_2 + \ldots + x_{t+1} = s, \quad x_1, x_2, \ldots, x_{t+1} \in \mathbb{N} \setminus \{0\}.$$  

This number is $\binom{s-1}{t}$. By summing over all $s \leq n$ and $t$ we get the required $O(2^n)$ bound.

5 $n^{O(1)} \cdot 2^{n/2}$-time algorithm for energy games

In this section we give an algorithm establishing Theorem 3.

We consider an energy game $G$ on a graph $G = (V, E)$ with a weight function $w: E \to \mathbb{R}$ and with a partition of $V$ between the players given by the sets $V_{\text{Max}}$ and $V_{\text{Min}}$. We assume that $G$ has $n$ nodes and $m$ edges.

First, we notice that without loss of generality we may assume that $G$ is bipartite.

Lemma 7. An energy game on $n$ nodes can be reduced in strongly polynomial time to a bipartite energy game on at most $n$ nodes.

This fact seems to be overlooked in the literature. Here is a brief sketch of it. Suppose that the pebble is in $a \in V_{\text{Max}}$. After controlling the pebble for some time Max might decide to enter a Min’s node $b$. Of course, it makes sense to do it via a path of the largest weight (among all paths from $a$ to $b$ with intermediate nodes controlled by Max). We can simply replace this path by a single edge from $a$ to $b$ of the same weight. Similar thing can be done with Min, but now the weight of a path should be minimized. By performing this for all pair of nodes controlled by different players we obtain an equivalent bipartite game. A full proof is given in Appendix B.

To simplify an exposition we first present our algorithm for the case when the following assumption is satisfied.

Assumption 1. In the graph $G$ there are no zero cycles.

Discussion of the general case is postponed to the end of this section.

Exposition of the algorithm follows the same scheme as for discounted games. First we define a polyhedron that we will work with. Now we call it polyhedron of potentials. In the algorithm we iterate the points of this polyhedron via valid shifts. To produce valid shifts we again use discounted normal play games. We should also modify a terminating condition. Given a point of polyhedron of potentials satisfying our new terminating condition one should be able to find all the nodes where Max wins in energy game. We also describe an analog of procedure RealizeGraph (again used to control the bit-length of points that arise in the algorithm). All this is collected together in the Algorithm 2. Here are details.

The polyhedron of potentials is defined as follows:

$$w(e) + x_b - x_a \leq 0 \quad \text{for } (a, b) \in E, a \in V_{\text{Max}},$$  

$$w(e) + x_b - x_a \geq 0 \quad \text{for } (a, b) \in E, a \in V_{\text{Min}}.$$  

(17)  

(18)
Here \( x \) is an \( n \)-dimensional real vector with coordinates indexed by the nodes of the graph. This polyhedron is denoted by \( \text{PolPoten} \).

By setting
\[
x_a = \begin{cases} 
W & a \in V_{\text{Max}}, \\
0 & a \in V_{\text{Min}},
\end{cases}
\]
for \( W = \max_{e \in E} |w(e)| \) we obtain that \( \text{PolPoten} \) is not empty (here it is important that our energy game is bipartite).

We use notions similar to those we gave for the optimality polyhedron. Namely, we call an edge \( e = (a, b) \in E \) tight for \( x \in \text{PolPoten} \) if \( w(e) + x_b - x_a = 0 \). The set of all \( e \in E \) that are tight for \( x \in \text{PolPoten} \) is denoted by \( E_x \). By \( G_x \) we mean the graph \((V, E_x)\). A very important consequence of the Assumption \( 1 \) is that for every \( x \in \text{PolPoten} \) the graph \( G_x \) is a directed acyclic graph. Indeed, a cycle consisting of edges that are tight for \( x \) would be a zero cycle, contradicting Assumption \( 1 \).

Next, we call a vector \( \delta \in \mathbb{R}^n \) a valid shift for \( x \in \text{PolPoten} \) if for all small enough \( \varepsilon > 0 \) it holds that \( x + \varepsilon \delta \in \text{PolPoten} \). Again, discounted normal play games on \( G_x \) can be used to produce a valid shift for \( x \). Now the discounted factor in a discounted normal play game is irrelevant. We can pick an arbitrary one, say, \( \lambda = 1/2 \). As before, for \( x \in \text{PolPoten} \) we let \( \delta_x \) be the solution to (4–7) for the graph \( G_x \). Since the graph \( G_x \) is acyclic, we have \( \delta_x(a) \neq 0 \) for every \( a \in V \). Define \( V^+_x = \{ a \in V \mid \delta_x(a) > 0 \} \) and \( V^-_x = \{ a \in V \mid \delta_x(a) < 0 \} \).

**Lemma 8.** Assume that \( x \in \text{PolPoten} \) and let \( \chi^+_x \) be the characteristic vector of the set \( V^+_x \). Then \( \chi^+_x \) is a valid shift for \( x \).

**Proof.** Assume that \( (a, b) \in E_x \). It is enough to show that \( \chi^+_x(b) - \chi^+_x(a) \leq 0 \) if \( a \in V_{\text{Max}} \) and \( \chi^+_x(b) - \chi^+_x(a) > 0 \) if \( a \in V_{\text{Min}} \).

First, assume that \( a \in V_{\text{Max}} \) and \( \chi^+_x(b) - \chi^+_x(a) > 0 \). Then \( \chi^+_x(a) = 0 \) and \( \chi^+_x(b) = 1 \), i.e., \( \delta_x(b) > 0 \) and \( \delta_x(a) < 0 \). But this contradicts (6).

Similarly, assume \( a \in V_{\text{Min}} \) and \( \chi^+_x(b) - \chi^+_x(a) < 0 \). Then \( \chi^+_x(a) = 1 \) and \( \chi^+_x(b) = 0 \), i.e., \( \delta_x(b) < 0 \) and \( \delta_x(a) > 0 \). This contradicts (7). \( \square \)

The following lemma specifies and justifies our new terminating condition.

**Lemma 9.** Let \( x \in \text{PolPoten} \) and assume that in the graph \( G \) there are no edges from \( V^+_x \cap V_{\text{Min}} \) to \( V^-_x \) and no edges from \( V^-_x \cap V_{\text{Max}} \) to \( V^+_x \). Then \( V^+_x \) is the set of nodes where \( \text{Max} \) wins in the energy game and \( V^-_x \) is the set of nodes where \( \text{Min} \) wins in the energy game.

**Proof.** Consider a positional strategy \( \sigma \) of \( \text{Max} \) defined as follows. For all \( a \in V^+_x \cap V_{\text{Max}} \) strategy \( \sigma \) goes from \( a \) by an edge \( (a, b) \in E_x \) with \( b \in V^+_x \). There is always such an edge because of (6) and because there are no sinks from \( V_{\text{Max}} \) in \( V^+_x \). In the nodes from \( V^-_x \cap V_{\text{Max}} \) define strategy \( \sigma \) arbitrarily.

Let us also define the following positional strategy \( \tau \) of \( \text{Min} \). For all \( a \in V^-_x \cap V_{\text{Min}} \) strategy \( \tau \) goes from \( a \) by an edge \( (a, b) \in E_x \) with \( b \in V^-_x \). Again, such an edge exists by
there are no zero cycles, so in every cycle consisting of nodes from $V_x^-$, the winning strategy is defined arbitrarily.

First, let us verify that for every $a \in V_x^+$ from $a$ one can reach only non-negative cycles in the graph $G^\sigma$. This would mean that Max wins in energy game from any node of $V_x^+$. First, in $G^\sigma$ from $a$ it is impossible to reach $V_x^-$. Indeed, $\sigma$ does not leave $V_x^+$ and by assumptions of the lemma there are no edges from $V_x^+ \cap V_{\text{Min}}$ to $V_x^-$. Hence it is enough to show that in the graph $G^\sigma$ every cycle consisting of nodes from $V_x^+$ is non-negative. Note that we can compute the weight of a cycle by summing up $w(e) + x_b - x_a$ over all edges $e = (a, b)$ belonging to a cycle (the terms $x_a$ cancel out). In turn, for edges of $G^\sigma$ lying inside $V_x^+$ all expressions $w(e) + x_b - x_a$ are non-negative. Indeed, for every $e$ that starts in $V_{\text{Min}}$ the expression $w(e) + x_b - x_a$ is non-negative by (18). In turn strategy $\sigma$ uses edges of the graph $G_x$, i.e., edges that are tight for $x$. For these edges we have $w(e) + x_b - x_a = 0$.

Similarly one can show that for every $a \in V_x^-$ from $a$ one can reach only non-positive cycles in the graph $G_x$. In fact, by Assumption 1 there are no zero cycles, so in every node from $V_x^-$ the winner of energy game is Min.

We define the procedure $\text{RealizeGraph}(S)$ similarly. The input to $\text{RealizeGraph}(S)$ is a subset $S \subseteq E$ and the output is a point $x \in \text{PolPoten}$ satisfying $S \subseteq E_x$, provided such point exists. Again, $\text{RealizeGraph}(S)$ can be computed in strongly polynomial time. Let us remark that now there is no need to refer to Megiddo’s algorithm [20]. Indeed, notice that all inequalities in (17–18) are of the form $x \leq y + c$, where $x$ and $y$ are variables and $c$ is a constant. It is clear that all inequalities appearing in Fourier–Motzkin elimination for (17–18) will still have this form. Hence we can keep the number of inequalities to be $O(n^2)$ throughout Fourier–Motzkin elimination, by removing redundant inequalities.

Now we are ready to give an algorithm establishing Theorem 3. Our goal is to find the sets

$$W_{\text{Max}} = \{a \in V \mid \text{Max wins from } a \text{ in the energy game}\},$$
$$W_{\text{Min}} = \{a \in V \mid \text{Min wins from } a \text{ in the energy game}\}.$$

\textbf{Algorithm 2: $n^{O(1)} \cdot 2^{n/2}$-time algorithm for energy games}

\begin{itemize}
  \item [**Result:**] The sets $W_{\text{Max}}, W_{\text{Min}}$.
  \item [**Initialization:**] $x = \text{RealizeGraph}(\emptyset)$;
  \item [**while**] there is an edge of $G$ from $V_x^+ \cap V_{\text{Min}}$ to $V_x^-$ or from $V_x^- \cap V_{\text{Max}}$ to $V_x^+$ do
    \begin{itemize}
      \item $\varepsilon_{\text{max}} \leftarrow$ the largest $\varepsilon \in (0, +\infty)$ s.t. $x + \varepsilon \chi^+_x \in \text{PolPoten}$;
      \item $x \leftarrow \text{RealizeGraph}(E_{x+\varepsilon_{\text{max}}\chi^+_x})$;
    \end{itemize}
  \item [**end**]
  \item output $W_{\text{Max}} = V_x^+$, $W_{\text{Min}} = V_x^-$;
\end{itemize}

The correctness of the output of our algorithm follows from Lemma 9. To compute $V_x^+, V_x^-$ and $\chi^+_x$ we find $\delta_x$ in strongly polynomial time by Lemma 4. In turn, we compute $\varepsilon_{\text{max}}$ in the same way as in Algorithm 1. To demonstrate the correctness of the algorithm
it only remains to show that $\varepsilon_{max} < +\infty$ throughout the algorithm. Indeed, when the terminating condition is not yet satisfied, there exists an edge $e = (a, b)$ of the graph $G$ such that either $a \in V_x^+ \cap V_{\text{Min}}, b \in V_x^- \cap V_{\text{Max}}$, or $a \in V_x^- \cap V_{\text{Max}}, b \in V_x^+$. Let us consider the first case, the second one is similar. Note that $(a, b)$ is not tight for $x$, because otherwise $(a, b)$ belongs to the graph $G_x$. This contradicts (7), because we can not have an edge from a Min’s node with positive value of $\delta_x$ to a node with negative value of $\delta_x$. So we have

$$w(e) + x_b - x_a > 0.$$ 

In turn, if we consider the left-hand side of the same inequality for $x + \varepsilon \chi^+_x$, we obtain the following:

$$w(e) + x_b - x_a + \varepsilon (\chi^+_x(b) - \chi^+_x(a)) = w(e) + x_b - x_a - \varepsilon.$$ 

Indeed, $\chi^+_x(a) = 1$ and $\chi^+_x(b) = 0$. This means that for some positive $\varepsilon$ the inequality corresponding to $(a, b)$ in (17-18) is tight for $x + \varepsilon \chi^+_x$. The same inequality, as we established, is not tight for $x$. Hence it is impossible to move along $\chi^+_x$ forever, i.e., $\varepsilon_{max} < +\infty$.

5.1 What if Assumption 1 does not hold?

Assume that we add small $\rho > 0$ to weights of all the edges. Then all non-negative cycles in $G$ become strictly positive. On the other hand, if $\rho$ is small enough, then all negative cycles stay negative. Thus, for all small enough $\rho > 0$ we obtain in this way an energy game equivalent to the initial one and satisfying Assumption 1. The problem is how to find $\rho > 0$ small enough so that this argument work.

If edge weights are rational numbers with bit-length at most $k$, then we can set $\rho = 2^{-k}/(n + 1)$. An interesting question is whether a suitable $\rho > 0$ can be found in strongly polynomial time. We do not know the answer. Instead, we propose another approach that solves energy games in strongly $n^{O(1)} \cdot 2^{n/2}$-time in general case.

Our idea is to add $\rho$ to all weights of edges not as a real number but as a formal variable. I.e., we will consider the weights as formal linear combinations of the form $a + b \cdot \rho, a, b \in \mathbb{R}$. First, we will perform additions over such combinations. More specifically, the sum of $a + b \cdot \rho$ and $c + d \cdot \rho$ will be $(a + c) + (b + d) \cdot \rho$. We will also perform comparisons of these linear combinations. We say that $a + b \cdot \rho < c + d \cdot \rho$ if $a < c$ or $a = c, b < d$. Note that the inequality $a + b \cdot \rho < c + d \cdot \rho$ holds for formal linear combinations $a + b \cdot \rho$ and $c + d \cdot \rho$ if and only if for all small enough $\gamma \in \mathbb{R}, \gamma > 0$ the same inequality holds for real numbers when one substitutes $\gamma$ instead of $\rho$.

Thus, more formally, we consider the weights as elements of the additive group $\mathbb{R}^2$ equipped with lexicographic order. Now, given our initial “real” energy game, we consider another one where the weight of an edge $e \in E$ is a formal linear combination $w(e) + \rho$. After that Assumption 1 is satisfied (again, if one understands the weight of a cycle as an element of the group $\mathbb{R}^2$).

We then run Algorithm 2, but now with the coordinates of the vector $x$ being elements of the group $\mathbb{R}^2$. Note that in Algorithm 2 we perform only additions and comparisons.
with the weights of edges and with the coordinates of \( x \). Indeed, in computing \( \varepsilon_{\text{max}} \) we solve at most \( m \) one-variable linear equations with the coefficient before the variable being 1. In computing \textit{RealizeGraph} procedure we perform Fourier–Motzkin elimination for inequalities of the form \( x \geq y + c \). Clearly, this also requires only additions and comparisons. So throughout the algorithm we never have to multiply or divide our formal linear combinations.

To argue that a version of Algorithm 2 with formal linear combinations is correct we use a sort of compactness argument. Fix some \( N \) and “freeze” the algorithm after \( N \) steps. Up to now only finitely many comparisons of linear combinations over \( \rho \) are performed. For all small enough real \( \gamma > 0 \) all these comparisons will have the same result if one substitutes \( \gamma \) instead of \( \rho \). So after \( N \) steps the “formal” version of Algorithm 2 will be in the same state as the “real” one, i.e., one where in advance we add a small enough real number \( \gamma \) to all the weights. In turn, for all small enough \( \gamma \) the “real” version terminates in \( N = n^{O(1)}2^{n/2} \) steps (see the next section) with the correct output to our initial energy game. It is important to note that a bound \( N \) on the number of steps of the “real” algorithm is independent of \( \gamma \). Hence the “formal” version also terminates in at most \( N = n^{O(1)}2^{n/2} \) steps with the correct output.

6 Energy games: complexity analysis

The complexity analysis of Algorithm 2 follows the same scheme as for discounted games. First, we define strong DNP games iteration (a more restrictive version of DNP games iteration, see Subsection 6.1). Then we consider a sequence \( x_0, x_1, x_2, \ldots \) of points from \textit{PolPoten} that arise in Algorithm 2. We show that the corresponding sequence of graphs \( G_{x_0}, G_{x_1}, G_{x_2}, \ldots \) can be obtained in strong DNP games iteration (Subsection 6.2). Finally, we show that the length of a strong DNP games iteration is bounded by \( O(2^{n/2}) \) (Subsection 6.3).

6.1 Definition of strong DNP games iteration

In strong DNP games iteration all graphs are assumed to be bipartite and acyclic.

Consider a directed bipartite acyclic graph \( H = (V, E_1) \). We say that a pair of nodes \( (a, b) \in V \times V \) is strongly improving for \( H \) if either \( a \in V_{\text{Min}}, \delta_H(a) > 0, \delta_H(b) < 0 \) or \( a \in V_{\text{Max}}, \delta_H(a) < 0, \delta_H(b) > 0 \). Here, as before, \( \delta_H \) is the solution to (4–7) for \( H \) (and for \( \lambda = 1/2 \)). Note once again that for acyclic graphs we have \( \delta_H(a) \neq 0 \) for all \( a \in V \).

Consider another directed bipartite acyclic graph \( K = (V, E_2) \) over the same set of nodes as \( H \). We say that \( K \) can be obtained from \( H \) in one step of strong DNP games iteration if \( E_2 \) contains all edges of \( H \) that are optimal for \( H \) and also at least one pair of nodes which is strongly improving for \( H \).

\footnote{Multiplications and divisions would not be a disaster for this argument as we could consider formal rational fractions over \( \rho \). However, we find it instructive to note that we never go beyond the group \( \mathbb{R}^2 \).}
Finally, we say that a sequence of directed bipartite acyclic graphs \( H_0, H_1, \ldots, H_j \) can be obtained in strong DNP games iterations if for all \( i \in \{0, 1, \ldots, j-1\} \) the graph \( H_{i+1} \) can be obtained from \( H_i \) in one step of strong DNP games iteration.

6.2 Why the sequence \( G_{x_0}, G_{x_1}, G_{x_2}, \ldots \) can be obtained in strong DNP games iteration

Consider any two consecutive points \( x \) and \( x' = \text{RealizeGraph}(E_x + \varepsilon_{\max} \chi_x^\pm) \) of PolPoten from Algorithm 2. We shall show that the graph \( G_{x'} \) can be obtained from \( G_x \) in one step of strong DNP games iteration. First, note that both of these graphs are bipartite (because the underlying energy game is bipartite) and acyclic (because of Assumption 1). Set \( y = x + \varepsilon_{\max} \chi_x^+ \). As the graph \( G_{x'} \) contains all edges of the graph \( G_y \), it is enough to show the following

(a) all the edges of the graph \( G_x \) that are optimal for \( G_x \) are also in the graph \( G_y \);

(b) there is an edge of the graph \( G_y \) which is a strongly improving pair for the graph \( G_x \).

Proof of (a). Take any edge \((a, b)\) of the graph \( G_x \) which is optimal for \( G_x \). Clearly, the values of \( \delta_x(a) \) and \( \delta_x(b) \) are either both positive or both negative. Hence the shift \( \chi_x^+ \) increases both \( x_a \) and \( x_b \) by the same amount. This means that \((a, b)\) is still tight for \( y \), i.e., \((a, b)\) is an edge of \( G_y \).

Proof of (b). First, there exists an edge \( e = (a, b) \in E \) which belongs to the graph \( G_y \) and not to \( G_x \). Indeed, otherwise all edges that are tight for \( y \) were already tight for \( x \) and hence \( \varepsilon_{\max} \) can be increased. It is enough to show now that any edge \((a, b) \in E_y \setminus E_x\) is strongly improving for \( G_x \). Since \((a, b)\) is not tight for \( x \), we have:

\[
\begin{align*}
&\bullet \ w(e) + x_b - x_a < 0 \text{ if } a \in V_{\text{Max}}; \\
&\bullet \ w(e) + x_b - x_a > 0 \text{ if } a \in V_{\text{Min}}.
\end{align*}
\]

On the other hand, since \((a, b)\) is tight for \( y \), we have:

\[
w(e) + x_b - x_a + \varepsilon_{\max}(\chi_x^+(b) - \chi_x^+(a)) = 0.
\]

Hence \( \chi_x^+(b) - \chi_x^+(a) > 0 \) if \( a \in V_{\text{Max}} \) and \( \chi_x^+(b) - \chi_x^+(a) < 0 \) if \( a \in V_{\text{Min}} \). Consider the case \( a \in V_{\text{Max}} \), the case \( a \in V_{\text{Min}} \) is similar. Note that \( \chi_x^+(b) - \chi_x^+(a) > 0 \) implies that \( \chi_x^+(b) = 1 \) and \( \chi_x^+(a) = 0 \). I.e., \( \delta_x(a) < 0 \) and \( \delta_x(b) > 0 \). Since \( a \in V_{\text{Max}} \), this means that \((a, b)\) is strongly improving for \( G_x \).

6.3 \( O(2^{n/2}) \) bound on length of strong DNP games iteration

Note that strong DNP games iteration is a special case of DNP games iteration. Hence all the results we established for DNP games iteration can be applied here. Since we are
dealing with bipartite graphs, we already have the bound $O(2^n)$ proved in Subsection 4.4. The improvement from $2^n$ to $2^{n/2}$ will be obtained by noticing that in every step of strong DNP games iteration both $f^H$ and $g^H$ increase in the alternating lexicographic order. Before we could only show that one of these vectors increase, the other could remain unchanged. So why the fact that both $f^H$ and $g^H$ increase each time leads to a $O(2^{n/2})$ bound? Note that both $f^H$ and $g^H$ are at least $\parallel f^H \parallel_1 = |\{a \in V \mid \delta_H(a) > 0\}|$ and $\parallel g^H \parallel_1 = |\{a \in V \mid \delta_H(a) < 0\}|$. Hence $\parallel f^H \parallel_1 + \parallel g^H \parallel_1 = n$. Therefore, if strong DNP games iteration has length $l$, then either $\parallel f^H \parallel_1 < n/l$ at least $l/2$ times or $\parallel g^H \parallel_1 < n/l$ at least $l/2$ times. Hence there are at least $l/2$ different vectors $v \in \mathbb{N}^{2n-1}$ satisfying (13–16) and $\parallel v \parallel_1 < n/l$. On the other hand, the number of such vectors is $O(2^{n/2})$. Indeed, as shown in Subsection 4.4 the number of $v \in \mathbb{N}^{2n-1}$ satisfying (13–16) and $\parallel v \parallel_1 = s, t(v) = t$ is $\left(\frac{s-1}{l}\right)$. By summing over all $s < n/l$ and $t$ we get the required $O(2^{n/2})$ bound.

It only remains to explain why both $f^H$ and $g^H$ increase in each step of a strong DNP games iteration. Let $H$ and $K$ be two consecutive graphs in strong DNP games iteration. Assume first that $f^K$ is not greater than $f^H$ in the alternating lexicographic order. By Lemma 5 for every $i \in \{0, 1, \ldots, n-1\}$ it holds that $\{a \in V \mid \delta_H(a) = \lambda_i^+\} = \{a \in V \mid \delta_K(a) = \lambda_i^+\}$. In particular, $\{a \in V \mid \delta_H(a) > 0\} = \{a \in V \mid \delta_K(a) > 0\}$. Since $\delta_H$ and $\delta_K$ are non-zero in all nodes (again, this is because these graphs are acyclic), we also have $\{a \in V \mid \delta_H(a) < 0\} = \{a \in V \mid \delta_K(a) < 0\}$. Hence a pair $(a, b) \in V \times V$ is strongly improving for $H$ if and only if it is strongly improving for $K$. On the other hand, the graph $K$ contains as an edge a strongly improving pair for $H$. This pair is also strongly improving for $K$. Therefore it can not be an edge of $K$, contradiction.

Exactly the same argument shows that $g^K$ is greater than $g^H$ in the alternating lexicographic order.

References

[1] Björklund, H., and Vorobyov, S. Combinatorial structure and randomized subexponential algorithms for infinite games. *Theoretical Computer Science* 349, 3 (2005), 347–360.

[2] Bojańczyk, M., and Czerwiński, W. An automata toolbox. A book of lecture notes, available at https://www.mimuw.edu.pl/~bojan/upload/reduced-may-25.pdf, 2018.

[3] Bouyer, P., Fahrenberg, U., Larsen, K. G., Markey, N., and Srba, J. Infinite runs in weighted timed automata with energy constraints. In *International Conference on Formal Modeling and Analysis of Timed Systems* (2008), Springer, pp. 33–47.

[4] Brim, L., Chaloupka, J., Doyen, L., Gentilini, R., and Raskin, J.-F. Faster algorithms for mean-payoff games. *Formal methods in system design* 38, 2 (2011), 97–118.
[5] Chakrabarti, A., De Alfaro, L., Henzinger, T. A., and Stoelinga, M. Resource interfaces. In *International Workshop on Embedded Software* (2003), Springer, pp. 117–133.

[6] Dorfman, D., Kaplan, H., and Zwick, U. A faster deterministic exponential time algorithm for energy games and mean payoff games. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)* (2019), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[7] Ehrenfeucht, A., and Mycielski, J. Positional strategies for mean payoff games. *International Journal of Game Theory* 8, 2 (1979), 109–113.

[8] Fijalkow, N., Gawrychowski, P., and Ohlmann, P. The complexity of mean payoff games using universal graphs. *arXiv preprint arXiv:1812.07072* (2018).

[9] Filar, J., and Vrieze, K. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

[10] Gimbert, H., and Zielonka, W. Games where you can play optimally without any memory. In *International Conference on Concurrency Theory* (2005), Springer, pp. 428–442.

[11] Halman, N. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all lp-type problems. *Algorithmica* 49, 1 (2007), 37–50.

[12] Hansen, T. D., and Ibsen-Jensen, R. The complexity of interior point methods for solving discounted turn-based stochastic games. In *Conference on Computability in Europe* (2013), Springer, pp. 252–262.

[13] Hansen, T. D., Miltersen, P. B., and Zwick, U. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM (JACM)* 60, 1 (2013), 1–16.

[14] Howard, R. A. Dynamic programming and markov processes.

[15] Jurdiński, M. Deciding the winner in parity games is in up∩co-up. *Information Processing Letters* 68, 3 (1998), 119–124.

[16] Lifshits, Y. M., and Pavlov, D. S. Potential theory for mean payoff games. *Journal of Mathematical Sciences* 145, 3 (2007), 4967–4974.

[17] Littman, M. L. *Algorithms for sequential decision making*. 1996.

[18] Ludwig, W. A subexponential randomized algorithm for the simple stochastic game problem. *Information and computation* 117, 1 (1995), 151–155.

[19] Matoušek, J., Sharir, M., and Welzl, E. A subexponential bound for linear programming. *Algorithmica* 16, 4-5 (1996), 498–516.
A Why Lemma 6 is tight

For $v \in A$ let $k(v)$ be the number of $i \in \{1, 2, \ldots, t(v)\}$ such that either $v_{2i} = 0$ or $v_{2i+1} = 0$. It is not hard to see that the number of $v \in A$ with $\|v\|_1 = n, t(v) = t, k(v) = k$ is

$$2^k \cdot \binom{t}{k} \cdot \binom{n-1}{2t-k}.$$  

(19)

Let us explain how to choose $t$ and $k$ so that (19) equals $(2 + \sqrt{2})^n$ (up to a polynomial factor). First, by using an approximation of binomial coefficients in terms of the Shannon function we get that up to a polynomial factor (19) equals:

$$2^{(\alpha \beta + \alpha h(\beta) + h(2\alpha - \alpha \beta))(n-1)},$$  

(20)

where $\alpha, \beta \in [0,1]$ are such that $t = \alpha (n-1)$ and $k = \beta t$, and $h(x) = x \log_2(1/x) + (1 - x) \log_2(1/(1 - x))$. A direct calculation shows that for

$$\alpha = \frac{\sqrt{2} + 1}{4}, \quad \beta = 2(\sqrt{2} - 1),$$

the coefficient before $(n - 1)$ in the exponent of (20) equals $\log_2(2 + \sqrt{2})$. This means that indeed (20), as well as (19), can be as large as $(2 + \sqrt{2})^n$ (up to a polynomial factor).

B Proof of Lemma 7

Let us call a node $a \in V$ of the graph $G$ trivial in the following two cases:
• $a \in V_{\text{Max}}$ and only nodes of $V_{\text{Max}}$ are reachable from $a$;
• $a \in V_{\text{Min}}$ and only nodes of $V_{\text{Min}}$ are reachable from $a$.

Next, let us call a cycle $C$ of the graph $G$ trivial in the following two cases:
• cycle $C$ is non-negative and all its nodes are from $V_{\text{Max}}$;
• cycle $C$ is negative and all its nodes are from $V_{\text{Min}}$.

First step of our reduction is to get rid of trivial nodes and cycles. Note that once we have detected a trivial node or a trivial cycle, we can determine the winner of energy game in at least one node of $G$. Indeed, to determine the winner of energy game in a trivial node we essentially need to solve a one-player energy game. It is well-known that this can be done in strongly polynomial time. In turn, all nodes of a trivial cycle are winning for the player controlling these nodes – he can win just by staying on the cycle forever.

Next, once the winner is determined in at least one node, there is a standard way of reducing the initial game to a game with fewer nodes. Suppose we know the winner in a node $a$, say, it is Max. Then Max also wins in all the nodes from where he can enforce reaching $a$. We simply remove all these nodes. This does not affect who wins the energy games in the remaining nodes. Indeed, Max has no edges to removed nodes and a winning strategy of Min would never use an edge to these nodes. It should be also noted that in the remaining graph all the nodes still have at least one out-going edge (a sink would have been removed).

So getting rid of trivial nodes and cycles can be done as follows. We first detect whether they exist. Then we determine the winner in some node of the graph and reduce our game to a game with smaller number of nodes. Clearly, all these actions take strongly polynomial time. This can be repeated at most $n$ times, so the whole procedure takes strongly polynomial time.

From now we assume that we are given an energy game $G$ on a graph $G = (V, E)$ with no trivial cycles and nodes. We construct a bipartite graph $G'$ over the same set of nodes and the corresponding bipartite energy game $G'$ equivalent to the initial one. In the definition of $G'$ we use the following notation. Consider a path $p$ of the graph $G$. We say that $p$ is Max-controllable if all the nodes of $p$ except the last one are from $V_{\text{Max}}$ (the last one can belong to $V_{\text{Min}}$ as well as to $V_{\text{Max}}$). In other words, Max should be able to navigate the pebble along $p$ without giving the control to Min. Similarly, we say that $p$ is Min-controllable if all the nodes of $p$ except the last one are from $V_{\text{Min}}$.

First, consider a pair of nodes $a \in V_{\text{Max}}, b \in V_{\text{Min}}$. We include $(a, b)$ as an edge to the graph $G'$ if and only if in $G$ there is a Max-controllable path from $a$ to $b$. Since $a$ is not a trivial node in $G$, there will be at least one edge starting at $a$ in $G'$. Provided $(a, b)$ was included, we let its weight in $G'$ be the largest weight of a Max-controllable path from $a$ to $b$ in $G$ (with respect to the weight function of $G$). We call a path on which this maximum is attained underlying for edge $(a, b)$. In this way we always obtain a finite weight since in $G$ there are no positive cycles consisting entirely of nodes from $V_{\text{Max}}$. 
We have described edges of $G'$ from $V_{\text{Max}}$ to $V_{\text{Min}}$. Edges in opposite direction are defined analogously. Namely, consider a pair of nodes $a \in V_{\text{Min}}, b \in V_{\text{Max}}$. We include this pair to $G'$ as an edge if and only if in $G$ there is a Min-controllable path from $a$ to $b$. Once $(a, b)$ is included, we let its weight be the minimal weight of a Min-controllable path from $a$ to $b$ in $G$. A path attaining this minimum will be called underlying for $(a, b)$. Again, absence of trivial nodes guaranties that in $G'$ the node $a$ will have at least one out-going edge. The weight of $(a, b)$ will be well-defined due to absence of trivial cycles.

It only remains to argue that $G'$ is equivalent to $G$. Let $W_{\text{Max}}$ ($W_{\text{Min}}$) be the set of nodes where Max (Min) wins in $G$. It is enough to show that the set $W_{\text{Max}}$ (the set $W_{\text{Min}}$) is winning for Max (Min) in $G'$. We present an argument only for $W_{\text{Max}}$, the argument for $W_{\text{Min}}$ is similar.

Let $\sigma$ be a Max’s positional strategy which is winning for the game $G$ in $W_{\text{Max}}$. Consider the following Max’s positional strategy $\sigma'$ for the graph $G'$. We will define it only for nodes in $W_{\text{Max}}$. Given a Max’s node $a \in W_{\text{Max}}$, apply $\sigma$ to $a$ repeatedly until a node from $V_{\text{Min}}$ is reached. In fact, there is a possibility that from a strategy $\sigma$ loops before reaching any Min’s node. But then the corresponding cycle would be negative (there are no trivial cycles). This would mean that $\sigma$ is not winning for Max in $a$. So we conclude that indeed by applying repeatedly $\sigma$ to $a$ we reach a node from $V_{\text{Min}}$. Let this node from $V_{\text{Min}}$ be $b$. Note that $(a, b)$ is an edge of $G'$, as we have reached $b$ by a Max-controllable path from $a$. We let $(a, b)$ be the edge that strategy $\sigma'$ uses in the node $a$.

We shall prove that only non-negative cycles are reachable from $W_{\text{Max}}$ in $(G')^{\sigma'}$. First, note that edges that $\sigma'$ uses do not leave $W_{\text{Max}}$. This is because by applying a winning Max’s strategy repeatedly we can not leave $W_{\text{Max}}$ in $G$. Moreover, no Min’s edge in $G'$ can leave $W_{\text{Max}}$. Indeed, otherwise Min could leave $W_{\text{Max}}$ in $G$. Thus, it remains to argue that any cycle $C'$ in $(G')^{\sigma'}$, located in $W_{\text{Max}}$, is non-negative. Indeed, we can obtain in $G^{\sigma}$ a cycle $C$, located in $W_{\text{Max}}$ and having at most the same weight. As $C$ is non-negative, the same holds for $C'$.

To obtain $C$ we replace each edge $(a, b)$ of $C'$ by a path $p_{(a,b)}$ from $a$ to $b$ in $G^{\sigma}$. The path $p_{(a,b)}$ will never leave $W_{\text{Max}}$ and its weight in $G$ will be at most the weight of $(a, b)$ in $G'$.

If $(a, b) \in V_{\text{Min}} \times V_{\text{Max}}$, we let $p_{(a,b)}$ be an underlying path for $(a, b)$. Its weight in $G$ just equals the weight of $(a, b)$ in $G'$. As this path is Min-controllable, it belongs to $G^{\sigma}$ and never leaves $W_{\text{Max}}$.

If $(a, b) \in V_{\text{Max}} \times V_{\text{Min}}$, then $(a, b)$ is used by strategy $\sigma'$ in $a$. Hence by definition of $\sigma'$ there is a Max-controllable path in $G^{\sigma}$ from $a$ to $b$. We let $p_{(a,b)}$ be this path. It never leaves $W_{\text{Max}}$ as $\sigma$ can not leave $W_{\text{Max}}$. The weight of $(a, b)$ in $G'$ is the largest weight of a Max-controllable path from $a$ to $b$ in $G$, so the weight of $p_{(a,b)}$ can only be smaller.