Faster Algorithm for Turn-based Stochastic Games with Bounded Treewidth

Krishnendu Chatterjee* Tobias Meggendorfer* Raimundo Saona* Jakub Svoboda*

Abstract

Turn-based stochastic games (aka simple stochastic games) are two-player zero-sum games played on directed graphs with probabilistic transitions. The goal of player-max is to maximize the probability to reach a target state against the adversarial player-min. These games lie in NP ∩ coNP and are among the rare combinatorial problems that belong to this complexity class for which the existence of polynomial-time algorithm is a major open question. While randomized sub-exponential time algorithm exists, all known deterministic algorithms require exponential time in the worst-case. An important open question has been whether faster algorithms can be obtained parametrized by the treewidth of the game graph. Even deterministic sub-exponential time algorithm for constant treewidth turn-based stochastic games has remain elusive. In this work our main result is a deterministic algorithm to solve turn-based stochastic games that, given a game with n states, treewidth at most t, and the bit-complexity of the probabilistic transition function $\log D$, has running time $O((tn^2 \log D)^{t \log n})$. In particular, our algorithm is quasi-polynomial time for games with constant or poly-logarithmic treewidth.

1 Introduction

Turn-based stochastic games. Turn-based stochastic games [12] are two-player zero-sum games played on a directed graph with probabilistic transitions. The state space (or vertex set) is partitioned into three types of states: player-max states, player-min states, and probabilistic states. At player-max and player-min states, the respective player chooses a successor state, and at probabilistic states a successor is chosen according to a given probability distribution. Strategies (or policies) for players provide the successor state choice at every player state, and fixing strategies for both players a Markov chain is obtained, for which probabilities are uniquely defined. We consider turn-based stochastic games where the goal of player-max is to reach a target state, and the adversary player-min has a complementary goal. These games are the perfect-information variant of games studied in the seminal works of Shapley [39] and Everett [22]. The special case of turn-based stochastic games where every state has at most two successors and the probabilities are uniform is called *simple stochastic games (SSGs)* [12], and turn-based stochastic games are polynomially equivalent to SSGs [12]. In the sequel, we simply write stochastic games to refer to turn-based stochastic games.

Value problem: complexity and algorithm. The value at a state of a stochastic game is the maximal probability that player-max can ensure to reach the target state irrespective of the strategy choice of player-min. The decision problem associated with the value (i.e., whether the value of a state is at least a given threshold) lies in NP ∩ coNP [12] (even UP ∩ coUP [7]). This is among the rare combinatorial problems that lie in NP ∩ coNP, yet the existence of a polynomial-time algorithm is a major and long-standing open problem. The classical algorithmic approaches to compute the values of a stochastic game are: (a) strategy improvement (or policy iteration) [11, 35]; (b) value-iteration [9]; and (c) reduction to non-convex non-linear optimization problem [11]. All the above approaches require exponential time in the worst case. There is a randomized sub-exponential time algorithm to compute values of stochastic games [34] that returns the value of each state in expected time of $2^{\tilde{\mathcal{O}}(\sqrt{n})}$, where n is the number of vertices.

Related game problems on directed graphs. There are interesting related problems on games on directed graphs without probabilistic transitions, namely, *parity games* [21] and *mean-payoff games* [20, 26, 42]. Parity games and mean-payoff games also lie in NP ∩ coNP [21] (even UP ∩ coUP [30]), and the existence of polynomial-time algorithms are major open problems. Parity games admit linear-time reduction to mean-payoff games [30] and mean-payoff games admit linear-time reduction to stochastic games [42]; however reductions in the converse direction are not known. For parity games several important algorithmic improvements have been achieved, in particular, deterministic sub-exponential time algorithm [32] and deterministic quasi-polynomial time algorithm [6], that break the exponential-time barrier exist. However no similar

*Institute of Science and Technology Austria.

algorithmic improvements have been achieved for mean-payoff games and stochastic games, and no deterministic sub-exponential time algorithm is known for mean-payoff or stochastic games.

Parametrized algorithms. Given the algorithmic improvements for mean-payoff and stochastic games have been rare, it is natural to consider faster parametrized algorithms. A very natural restriction to consider, both from theoretical and practical perspective, is game graphs with small treewidth. We first discuss the theoretical perspective. Many fundamental algorithmic questions in graph theory that are NP-complete have efficient algorithms parametrized by treewidth [4, 15]. In computational problems in logic, where parity, mean-payoff and stochastic games are relevant, the celebrated result of Courcelle [13] shows that Monadic Second Order (MSO) problems with constant treewidth admit efficient algorithms; and parity games with constant treewidth graphs can be solved in polynomial time [36]. From the practical perspective many problems in programming languages and formal methods naturally have small treewidth. For example, control-flow graphs of programs often have constant treewidth [40], and the constant treewidth property has been exploited for faster algorithms in programming languages [8] and formal verification [10].

Open problem. While special classes of stochastic games, namely, parity games have been studied with the constant treewidth property, faster algorithms that can exploit small treewidth have remain elusive for stochastic games (and even for mean-payoff games). The classical approaches such as value-iteration and strategy improvement require exponential time on constant treewidth stochastic games; and exploiting treewidth for non-convex non-linear optimization problem is quite challenging. In particular, whether there is a deterministic sub-exponential time algorithm for constant treewidth stochastic games (or even the special case of mean-payoff games) has been an important open question.

Our main result. In this work our main result which answers the above open question is as follows: We present a deterministic algorithm that, given a stochastic game with n states that have treewidth at most t and where the bit-complexity of the transition function is $\log D$, computes the value of all states in time $\mathcal{O}((tn^2 \log D)^{t \log n})$.

Implications. Our main result has several implications:

- For stochastic games with small treewidth which is constant or poly-logarithmic in n, our deterministic algorithm runs in quasi-polynomial time.
- For stochastic games with treewidth at most $\mathcal{O}(n^{1-\varepsilon})$, with $\varepsilon > 0$, our deterministic algorithm runs in sub-exponential time. Thus for a large class of stochastic games (which includes planar graphs with treewidth $\sqrt{n}$) we obtain a deterministic sub-exponential time algorithm.
- Our main result and the above results hold for mean-payoff games, where $\log D$ is replaced by $\log nW$, where W is the maximal absolute cost/reward in mean-payoff games.

Technical contribution. In this work we present a new approach to solve stochastic games based on guessing values and checking whether the guessed value is an upper or lower bound. In general the running time of this new approach is exponential in the worst case. However we show that this new approach can exploit separator properties of graphs to obtain faster algorithm for stochastic games with bounded treewidth.

Related works. The algorithmic study of stochastic games and its special classes as mean-payoff and parity games have received significant attention. Below we summarize some related works.

- The algorithmic study of stochastic games has been considered in several works [11, 12, 34]. Faster algorithms for stochastic games parametrized by the number of probabilistic states have been considered in [25, 29]. Variants of strategy improvement algorithms have also been considered recently [17]. However, these approaches do not present a sub-exponential time algorithm for small treewidth stochastic games.
- The algorithmic study of mean-payoff games has also been considered in several works [20, 26, 42, 5, 19]. All these algorithms have a worst-case complexity of exponential, and none of the algorithms is known to exploit treewidth to obtain sub-exponential time complexity.
- The algorithmic study of parity games has achieved significant progress. While the classical algorithms [21, 41] has exponential worst-case complexity, faster exponential-time algorithms were achieved [31, 38], and then deterministic sub-exponential [32] and deterministic quasi-polynomial time algorithms [6] were obtained. Moreover, efficient algorithms exist for parity games with small treewidth [36, 23], e.g., polynomial-time for constant treewidths. However, extending the algorithmic bounds for parity games to mean-payoff or stochastic games has been a major open question.

2 Preliminaries

For a finite set S we denote by $\mathcal{D}(S)$ the set of all probability distributions over S.

Turn-based stochastic games. A *(turn-based two-player) stochastic game* (SG) [39, 11] is a tuple $\mathsf{G} = (S, E, \Delta, (S_{\max}, S_{\min}, S_p, \{s_+, s_-\}))$ with the following components:

- S is a finite set of states partitioned into player-max's states $S_{\max}$, player-min states $S_{\min}$, probabilistic states S_p, and two special absorbing (or sink) states, namely, the target state s_+ and the avoid state s_-.
- The probabilistic transition function Δ assigns to every probabilistic state a probability distribution over the successor states, i.e., $\Delta\colon S_p \to \mathcal{D}(S)$ assigns to every $s \in S_p$ the probability $\Delta(s, s')$ of transition from state s to state s'.
- The edge set $E \subseteq S \times S$ and we write $E(s) = \{s' \mid (s, s') \in E\}$ and require the following properties: (a) $E(s)$ is non-empty for all states; (b) for all $s \in S_p$ we have $E(s) = \{s' \mid \Delta(s, s') > 0\}$; and (c) $E(s_+) = \{s_+\}$ and $E(s_-) = \{s_-\}$.

In the sequel, we write *game* to denote a stochastic game. We associate the graph $\widehat{G} = (S, E)$ as the graph of game G.

Strategies. Strategies (or policies) are recipes for players to choose the successor states. A (positional, aka. memoryless) strategy σ for player-max assigns to each state $s \in S_{\max}$ a successor $\sigma(s) \in E(s)$. Min-player strategies τ are defined analogously. A pair of strategies $\pi = (\sigma, \tau)$ is called *(strategy) profile*. For a profile $\pi = (\sigma, \tau)$ we define $\pi(s) = \sigma(s)$ if $s \in S_{\max}$ and $\tau(s)$ if $s \in S_{\min}$. We write Σ_{G}, Γ_{G}, and $\Pi_{\mathsf{G}} \coloneqq \Sigma_{\mathsf{G}} \times \Gamma_{\mathsf{G}}$ for the set of player-max's strategies, player-min strategies, and strategy profiles, respectively.

Probability space. Given a game G and a strategy profile $\pi = (\sigma, \tau)$ we have a Markov chain $\mathsf{G}[\pi] = (S, \Delta^\pi)$, where (i) $\Delta^\pi(s, s') \coloneqq 1$ if $s \in S_{\max} \cup S_{\min}$ and $\pi(s) = s'$; (ii) $\Delta^\pi(s, s') \coloneqq \Delta(s, s')$ for $s \in S_p$; and (iii) s_+, s_- are absorbing states. For every initial state $s \in S$, the Markov chain $\mathsf{G}[\pi]$ has associated a unique probability space $(\Omega, \mathcal{F}, \mathbb{P}^\pi_{\mathsf{G},s})$ where each element $\omega \in \Omega$ is an infinite path ω in the game G, which corresponds to an infinite walk in the graph $\widehat{G}$. The σ-algebra $\mathcal{F}$ is induced by the cone sets of Ω and an *event* is a measurable subset of Ω. Also, we simply write $\mathbb{P}^\pi_{\mathsf{G}}$ (instead of $\mathbb{P}^\pi_{\mathsf{G},s}$) if the initial state s is clear from context.

Reachability and Safety. We formally define reachability and safety events. Given a set $T \subseteq S$ of states, the reachability event $\mathrm{Reach}(s, T)$ from a starting state s is the set of all paths that at some point reach T, i.e., $\{\omega = (s_0, s_1, \ldots) \mid s_0 = s \text{ and } \exists i.\ s_i \in T\}$; and the safety event $\mathrm{Safe}(s, T)$ from a starting state s is the set of all paths that never visits states in T, i.e., $\{\omega = (s_0, s_1, \ldots) \mid s_0 = s \text{ and } \forall i.\ s_i \notin T\}$. For a singleton set $T = \{s'\}$ we simply write $\mathrm{Reach}(s, s')$ to denote $\mathrm{Reach}(s, \{s'\})$, and analogously for $\mathrm{Safe}(s, s')$. Also, we simply write $\mathrm{Reach}(T)$, instead of $\mathrm{Reach}(s, T)$ if the initial state s is clear from context, and analogously for $\mathrm{Safe}(T)$.

Problem Statement: (Value function computation). For a game G compute the value function $\mathsf{val}_{\mathsf{G}}\colon S \to [0, 1]$ that assigns to every state s the maximal probability that player-max can ensure to reach s_+. Formally,

$$\mathsf{val}_{\mathsf{G}}(s) \coloneqq \max_{\sigma \in \Sigma_{\mathsf{G}}} \min_{\tau \in \Gamma_{\mathsf{G}}} \mathbb{P}^{(\sigma,\tau)}_{\mathsf{G},s}(\mathrm{Reach}(s_+)) \,.$$

Stochastic games are *determined*, i.e. switching the order of $\max$ and $\min$ quantification over the strategies does not change the value [12] similar to the minimax theorem. A strategy σ (resp., τ) is *optimal* if for all states s we have $\mathsf{val}_{\mathsf{G}}(s) = \min_{\tau \in \Gamma_{\mathsf{G}}} \mathbb{P}^\pi_{\mathsf{G},s}(\mathrm{Reach}(s_+))$ (resp., $\mathsf{val}_{\mathsf{G}}(s) = \max_{\sigma \in \Sigma_{\mathsf{G}}} \mathbb{P}^\pi_{\mathsf{G},s}(\mathrm{Reach}(s_+))$), where $\pi = (\sigma, \tau)$.

One-step update function. Given a function $f\colon S \to \mathbb{R}$ assigning values to states we define $\widehat{f} \coloneqq \textsc{OneStep}(\mathsf{G}, f)$ to be the function from $S \to \mathbb{R}$ as follows:

$$\widehat{f}(s) \coloneqq \begin{cases} \max_{s' \in E(s)} f(s') & s \in S_{\max} \\ \min_{s' \in E(s)} f(s') & s \in S_{\min} \\ \sum_{s' \in S} \Delta(s, s') \cdot f(s') & s \in S_p \\ f(s) & s \in \{s_-, s_+\} \,. \end{cases}$$

This denotes the one-step update for every state other than these special states.

Some useful properties. We present some useful properties related to stochastic games.

1. *Fixpoint characterization.* The value function satisfies the fixpoint equation $\mathsf{val}_{\mathsf{G}} = \textsc{OneStep}(\mathsf{G}, \mathsf{val}_{\mathsf{G}})$. Moreover, $\mathsf{val}_{\mathsf{G}}$ is the *least* fixpoint of the equation [16].

2. *Optimal successors.* For a state $s \in S_{\max} \cup S_{\min}$, define $\mathsf{OptSucc}(s) \coloneqq \{s' \in E(s) \mid \mathsf{val}_{\mathsf{G}}(s) = \mathsf{val}_{\mathsf{G}}(s')\}$, denote the set of successor states that have the same value. The fixpoint characterization ensures that $\mathsf{OptSucc}(s)$ is non-empty for all $s \in S_{\max} \cup S_{\min}$.

3. *Optimal strategy characterization for player-min.* Note that player-max has a reachability objective and the objective for player-min is the complement safety objective. For safety objectives, locally optimal choices ensure an optimal strategy, i.e., consider a stategy τ such that for all $s \in S_{\min}$ we have $\tau(s) \in \mathsf{OptSucc}(s)$, then τ is an optimal strategy [1]. For reachability objectives, such characterization is not true.

3 Guessing Values in Stochastic Games

In this section, we present our new *guessing* approach to solve stochastic games. Before we present the algorithm we introduce the notion of *trivial* states.

Trivial state. We say a state $s \in S_p$ is *trivial* if its successors only contain s_+ and s_-, i.e., $E(s) \subseteq \{s_+, s_-\}$. In this case the value of s is $\Delta(s, s_+)$. A state that is not trivial is called non-trivial.

Intuition of the guessing algorithm. The intuitive description of the guessing algorithm is as follows. For an arbitrary, non-trivial state, we guess its value. Under the assumption that this guess is correct, we solve a simpler game with one less non-trivial state (we can apply standard procedures or use our method recursively). Based on the obtained solution, we then determine whether our guess was too small or too large. This allows a binary search on the value of a state approximating it to any arbitrary precision. Finally, a rounding procedure allows us to obtain the exact value function once sufficient precision is achieved through the binary search.

In general, this approach has exponential runtime similar to the state-of-the-art approaches. However, in Section 4, we show how the notion of *decomposition*, for example in *bounded treewidth* games, can be combined with our approach to obtain faster algorithms.

3.1 The Idea of Guessing Values

Throughout this section, we fix an arbitrary game G. Below, we introduce the notion of a "reduced game".

Reduced game. For a game G, a state s, and a guess $0 \leq \gamma \leq 1$ of its value, we denote $\mathsf{G}[s = \gamma]$ the reduced game obtained from G where s is replaced by a (trivial) probabilistic state with $\Delta(s, s_+) = \gamma$ and $\Delta(s, s_-) = 1 - \gamma$. Observe that successors of s in $\mathsf{G}[s = \gamma]$ are only target s_+ and avoid s_- and $\mathsf{val}_{\mathsf{G}[s=\gamma]}(s) = \gamma$.

Steps of the guessing procedure. The guessing procedure comprises of three steps: (i) assigning a value γ to a non-trivial state s, (ii) solving the reduced game $\mathsf{G}[s = \gamma]$, and (iii) determining whether the guessed value is an over- or under-approximation of the value of that state, based on the solution of the reduced game.

Intuitive overview of correctness. For the sake of explanation, assume that $s \in S_{\max}$ (s is a maximizer state). Recall that $\mathsf{val}_{\mathsf{G}[s=\gamma]}$ is the value function of the reduced game $\mathsf{G}[s = \gamma]$. Regarding the value function $\mathsf{val}_{\mathsf{G}[s=\gamma]}$ and the value $\mathsf{val}_{\mathsf{G}}(s)$ at state s there are two possible scenarios. Assume that there is a successor s' of s in G which achieves a value larger than γ in $\mathsf{G}[s = \gamma]$. Then, we prove that $\mathsf{val}_{\mathsf{G}}(s) > \gamma$ because in G, player-max can move to s' and preserve the strategy from $\mathsf{G}[s = \gamma]$. If all successors of s have a value at most γ in $\mathsf{G}[s = \gamma]$, possibly "relying" on the value γ of s in $\mathsf{G}[s = \gamma]$, then, by similar reasoning, we prove that the guess γ is an upper bound for $\mathsf{val}_{\mathsf{G}}(s)$. Our proof relies on a careful analysis of the runs that eventually reach the state s and those which do not.

3.2 Guess Verification

We formalize the key principle behind the guessing procedure in the following lemma.

LEMMA 3.1. *Consider a game G, a state $s \in S$ and a guess $\gamma \in [0, 1]$. Let $f \coloneqq \mathsf{val}_{\mathsf{G}[s=\gamma]}$ be the value function of the reduced game when s is assumed to have value γ. Let $\widehat{f} = \textsc{OneStep}(\mathsf{G}, f)$ and $\gamma' \coloneqq \widehat{f}(s)$. Then,*

$$\gamma' > \gamma \quad \Leftrightarrow \quad \mathsf{val}_{\mathsf{G}}(s) > \gamma\,.$$

Projected strategy π^s. To "establish correspondence" between the original game G and the reduced game $\mathsf{G}[s = \gamma]$, we define π^s as the profile π projected on $\mathsf{G}[s = \gamma]$, i.e., π with the decision in s removed (recall that in $\mathsf{G}[s = \gamma]$ the state s is a probabilistic state).

To determine whether a guess is a lower or an upper bound, we separate the set of all runs reaching the target into those visiting s on the way and those never visiting s. The following lemma formalizes this idea.

LEMMA 3.2. *For a game* G*, strategy profile* π*, and states* s*,* s'*, the paths paths starting on* s' *reaching* s_+ *can be splitted in those that visit* s *or not. Formally,*

$$\mathbb{P}^{\pi}_{s',\mathsf{G}}(\mathrm{Reach}(s_+)) = \mathbb{P}^{\pi}_{s',\mathsf{G}}(\mathrm{Reach}(s)) \cdot \mathbb{P}^{\pi}_{\mathsf{G},s}(\mathrm{Reach}(s_+)) + \mathbb{P}^{\pi}_{s',\mathsf{G}}(\mathrm{Reach}(s_+) \cap \mathrm{Safe}(s)) \,.$$

Proof. Follows from partitioning the event $\mathrm{Reach}(s', s_+)$. Indeed, since paths starting at s' can either eventually visit s or never visit it, we can partitio $\mathrm{Reach}(s', s_+)$ in the following dijoint union.

$$\mathrm{Reach}(s', s_+) = (\mathrm{Reach}(s', s_+) \cap \mathrm{Reach}(s', s)) \sqcup (\mathrm{Reach}(s', s_+) \cap \mathrm{Safe}(s', s)) \ .$$

Moreover, since the target s_+ is absorbing, the set of paths that start at s', visit s and then stay on s_+, formally $\mathrm{Reach}(s', s_+) \cap \mathrm{Reach}(s', s)$, can be decomposed in a prefix and suffix as follows. The prefix of the path starts with s' and ends with s, while the suffix of the path starts with s and ends with s_+. Therefore, since $\mathsf{G}[\pi]$ is a Markov chain, we have that

$$\mathbb{P}^{\pi}_{s',\mathsf{G}}(\mathrm{Reach}(s', s_+) \cap \mathrm{Reach}(s', s)) = \mathbb{P}^{\pi}_{s',\mathsf{G}}(\mathrm{Reach}(s', s)) \cdot \mathbb{P}^{\pi}_{\mathsf{G},s}(\mathrm{Reach}(s, s_+)) \,,$$

which proves the desired result by additivity of $\mathbb{P}^{\pi}_{s',\mathsf{G}}$. ∎

Also, by considering one step in the stochastic evolution in $\mathsf{G}[\pi]$, we deduce the following.

COROLLARY 3.1. *For a game* G*, strategy profile* π*, and state* s *we have*

$$\begin{aligned}\mathbb{P}^{\pi}_{\mathsf{G},s}(\mathrm{Reach}(s_+)) &= \sum_{s' \in S} \Delta^{\pi}(s, s') \cdot \mathbb{P}^{\pi}_{s',\mathsf{G}}(\mathrm{Reach}(s_+)) \\ &= \sum_{s' \in S} \Delta^{\pi}(s, s') \left(\mathbb{P}^{\pi}_{s',\mathsf{G}}(\mathrm{Reach}(s)) \cdot \mathbb{P}^{\pi}_{\mathsf{G},s}(\mathrm{Reach}(s_+)) + \mathbb{P}^{\pi}_{s',\mathsf{G}}(\mathrm{Reach}(s_+) \cap \mathrm{Safe}(s))\right) \,.\end{aligned}$$

Key intuition for Lemma 3.1. First, we use Lemma 3.2 to reason about strategies of player-max and their minimum reachability probabilities in G and $\mathsf{G}[s = \gamma]$. We use the fact that, for any $s' \in S$, the probability of any event contained in $\mathrm{Safe}(s', s)$ is the same in G and $\mathsf{G}[s = \gamma]$. If $\mathsf{val}_{\mathsf{G}}(s) > \gamma$, we take an optimal strategy of player-max that obtains $\mathsf{val}_{\mathsf{G}}$ in G and project it to $\mathsf{G}[s = \gamma]$. The strategy is good enough such that $\gamma' > \gamma$. On the other hand, if $\gamma' > \gamma$, that means in $\mathsf{G}[s = \gamma]$ exists a strategy for the max player that for some successor of s reaches the target with higher probability. We take the strategy and extend it with an optimal decision in s. We show that by this, we get $\mathsf{val}_{\mathsf{G}}(s) > \gamma$.

Proof. [Proof of Lemma 3.1] First, by the decomposition of runs formalized in Corollary 3.1, for any guess γ and profile π, we obtain the following expression for $\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s, s_+))$

$$\sum_{s' \in S} \Delta^{\pi}(s, s') \left(\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s)) \cdot (\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s, s_+)) - \gamma + \gamma) + \mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s_+) \cap \mathrm{Safe}(s', s))\right) \,, \tag{3.1}$$

by inserting $0 = \gamma - \gamma$. Consider the projected strategy π^s. Through Lemma 3.2 we also obtain

$$\mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+)) = \mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s)) \cdot \gamma + \mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s_+) \cap \mathrm{Safe}(s', s)) \tag{3.2}$$

since, on the game $\mathsf{G}[s = \gamma]$, the state s reaches the target s_+ with probability γ independently of the strategy profile, i.e. $\mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s, s_+)) = \gamma$. Also, the probability of, starting at s, reaching the target s_+ while avoiding s is the same in $\mathsf{G}[\pi]$ and $\mathsf{G}[s = \gamma, \pi^s]$, i.e. $\mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+) \cap \mathrm{Safe}(s', s)) = \mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s_+) \cap \mathrm{Safe}(s', s))$.

We rearrange Eq. (3.2) to obtain that

$$\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s_+) \cap \mathrm{Safe}(s', s)) = \mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+)) - \mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s)) \cdot \gamma \,.$$

Moreover, notice that the probability of eventually reaching s is independent of what happens after the state s is reached, namely $\mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s)) = \mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s))$. Therefore,

$$\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s_+) \cap \mathrm{Safe}(s', s)) = \mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+)) - \mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s)) \cdot \gamma \,.$$

Replacing $\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s_+) \cap \mathrm{Safe}(s', s))$ in Eq. (3.1), we get the following expression for $\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s, s_+))$

$$\sum_{s' \in S} \Delta^{\pi}(s, s') \left(\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s)) \cdot (\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s, s_+)) - \gamma) + \mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+)) \right) , \tag{3.3}$$

which is the key equation to link the value function of the reduced game $\mathsf{G}[s = \gamma]$ and the original game G.

Note that if σ is optimal in G, then, for any strategy of player-min τ, we have that $\mathbb{P}^{(\sigma,\tau)}_{\mathsf{G}}(\mathrm{Reach}(s, s_+)) \geq \mathsf{val}_{\mathsf{G}}(s)$. With this, we are ready to proceed with the main proof, showing each direction separately.

1. If $\mathsf{val}_{\mathsf{G}}(s) > \gamma$, then $\gamma' > \gamma$. Let σ be an optimal strategy for player-max, τ an arbitrary strategy for player-min and set $\pi = (\sigma, \tau)$. Then, we have $\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s, s_+)) \geq \mathsf{val}_{\mathsf{G}}(s) > \gamma$. We prove that $\gamma' > \gamma$ by showing that the projected strategy σ^s achieves strictly more than γ in $\mathsf{G}[s = \gamma]$.

   Since $\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s, s_+)) > \gamma$, and $\gamma \geq 0$, the strategy σ ensures that the target is reached with positive probability. Thus, starting from s, the probability of eventually returning to s is less than one, namely

   $$\sum_{s' \in S} \Delta^{\pi}(s, s') \mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s', s)) < 1 . \tag{3.4}$$

   Inserting Eq. (3.4) into Eq. (3.3) yields us

   $$\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s, s_+)) < \mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s, s_+)) - \gamma + \sum_{s' \in S} \Delta^{\pi}(s, s') \left(\mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+)) \right) ,$$

   and rearranging the terms, we get

   $$\gamma < \sum_{s' \in S} \Delta^{\pi}(s, s') \left(\mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+)) \right) ,$$

   which implies that σ^s achieves strictly more than γ in $\mathsf{G}[s = \gamma]$. To introduce γ', note that, by definition of the value function, for all profiles π^s where player-min's strategy is optimal and states s', we have that $\mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+)) \leq \mathsf{val}_{\mathsf{G}[s=\gamma]}(s')$. Therefore,

   $$\gamma < \sum_{s' \in S} \Delta^{\pi}(s, s') \left(\mathsf{val}_{\mathsf{G}[s=\gamma]}(s') \right) \leq \textsc{OneStep}(\mathsf{G}, \mathsf{val}_{\mathsf{G}[s=\gamma]})(s) ,$$

   where the last inequality follows from the definition of $\textsc{OneStep}(\mathsf{G}, \mathsf{val}_{\mathsf{G}[s=\gamma]})$. We conclude that

   $$\gamma < \textsc{OneStep}(\mathsf{G}, \mathsf{val}_{\mathsf{G}[s=\gamma]})(s) = \gamma' .$$

2. If $\gamma' > \gamma$, then $\mathsf{val}_{\mathsf{G}}(s) > \gamma$. Let σ be a player-max's strategy in G that is optimal in $\mathsf{G}[s = \gamma]$ and, if $s \in S_{\max}$, then σ is extended with $\sigma(s)$ defined as a best successor of s according to $\mathsf{val}_{\mathsf{G}[s=\gamma]}$. Let τ an arbitrary player-min's strategy for G and set $\pi = (\sigma, \tau)$. We prove that $\mathsf{val}_{\mathsf{G}}(s) > \gamma$ by essentially reconstructing the proof of the previous item backwards.

   Note that, since τ might not be optimal and s might be a minimizer state,

   $$\gamma' = \textsc{OneStep}(\mathsf{G}, \mathsf{val}_{\mathsf{G}[s=\gamma]})(s) \leq \sum_{s' \in S} \Delta^{\pi}(s, s') \cdot \mathsf{val}_{\mathsf{G}[s=\gamma]}(s') .$$

   Moreover, since σ^s is optimal in $\mathsf{G}[s = \gamma]$, for all states s' we have $\mathsf{val}_{\mathsf{G}[s=\gamma]}(s') \leq \mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+))$, yielding that

   $$\gamma < \gamma' \leq \sum_{s' \in S} \Delta^{\pi}(s, s') \cdot \mathbb{P}^{\pi^s}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s', s_+)) . \tag{3.5}$$

Recalling Eq. (3.3) and using Eq. (3.5), we get

$$\begin{aligned}\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s,s_+)) &= \sum_{s'\in S}\Delta^{\pi}(s,s')\left(\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s))\cdot(\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s,s_+))-\gamma)+\mathbb{P}^{\pi^{s}}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s',s_+))\right)\\ &= \sum_{s'\in S}\Delta^{\pi}(s,s')\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s))\cdot(\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s,s_+))-\gamma)\\ &\quad+\sum_{s'\in S}\Delta^{\pi}(s,s')\mathbb{P}^{\pi^{s}}_{\mathsf{G}[s=\gamma]}(\mathrm{Reach}(s',s_+))\\ &> \sum_{s'\in S}\Delta^{\pi}(s,s')\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s))\cdot(\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s,s_+))-\gamma)+\gamma\,,\end{aligned}$$

which we rearrange to

$$\left(1-\sum_{s'\in S}\Delta^{\pi}(s,s')\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s))\right)\cdot\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s,s_+))>\left(1-\sum_{s'\in S}\Delta^{\pi}(s,s')\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s))\right)\cdot\gamma\,.$$

By definition of the value function, $\mathsf{val}_{\mathsf{G}}(s)\geq\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s,s_+))$, so we get that

$$\left(1-\sum_{s'\in S}\Delta^{\pi}(s,s')\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s))\right)\cdot\mathsf{val}_{\mathsf{G}}(s)>\left(1-\sum_{s'\in S}\Delta^{\pi}(s,s')\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s))\right)\cdot\gamma\,, \tag{3.6}$$

and we only need to prove that

$$\sum_{s'\in S}\Delta^{\pi}(s,s')\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s))<1\,, \tag{3.7}$$

namely, under the strategy profile π, starting from s there is a positive probability of never coming back.

Since $\gamma<\gamma'$, there exists a successor s' of s such that $\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s_+))>\gamma$ and therefore $\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s))<1$ (otherwise $\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(s',s_+))$ would be equal to γ). Consequently, Eq. (3.7) holds. Finally, simplifying Eq. (3.6), we get that $\mathsf{val}_{\mathsf{G}}(s)>\gamma$.

The desired result follows. ∎

REMARK 3.1. *Note that if $\gamma'=\gamma$, then we can only conclude that $\mathsf{val}_{\mathsf{G}}(s)\leq\gamma$. This is because if $\gamma'=\gamma$ it can be guaranteed that we have obtained a fixpoint, but not necessarily the least fixpoint of* ONESTEP$(\mathsf{G},\cdot)$. *In our guessing procedure if $\gamma=\gamma'$ we only consider γ as an upper bound and proceed.*

We use the above principle to obtain an arbitrary approximation of the value of a state, given that we have access to an oracle that computes exactly the value function of a simpler game. In Section 3.4 we show how to derive an exact solver from a sufficiently-precise approximation of the value function.

3.3 Guessing as Approximate Solver In this section, we analyse an approximate solver for the *value problem*, based on guessing and an oracle to compute the exact values of a simpler game.

> **Problem Statement (Approximation of value function).** For a game G and $\varepsilon>0$, obtain $v\colon S\to[0,1]$ such that, for all $s\in S$, we have $|v(s)-\mathsf{val}_{\mathsf{G}}(s)|\leq\varepsilon$.

Informal description of Algorithm 1: Procedure APPROXSOLVEGUESS. In Line 4, we set lower and upper bounds as 0 and 1, respectively, which contains the value of the state. In Line 5, a standard binary search starts refining these bounds. In Line 6, the guess γ is defined as the midpoint of the valid interval. In Line 7, it obtains the value function of the reduced game $\mathsf{val}_{\mathsf{G}[s=\gamma]}$ through the oracle. In Line 8, then γ' is computed as ONESTEP update of s from $\mathsf{val}_{\mathsf{G}[s=\gamma]}$ in G. It is the optimal reachability one can achieve by performing one step in G and then continuing the play in $\mathsf{G}[s=\gamma]$. Finally, based on the relationship between the guess γ and γ' (recall Lemma 3.1), it computes the new valid interval, to continue the binary search if necessary.

Algorithm 1 Approximate solve by guessing

Input: Game G, state s, approximation error $\varepsilon > 0$, oracle for the value problem SOLVE
Output: $v \in [0,1]$ which is an ε-approximation of $\mathsf{val}_{\mathsf{G}}(s)$

procedure APPROXSOLVEGUESS(G, s, ε, SOLVE)
  if s is trivial **then**
    return $\Delta(s, s_+)$
  $[\ell, u] \leftarrow [0,1]$ ▷ Valid bounds on $\mathsf{val}_{\mathsf{G}}(s)$
  while $u - \ell > 2\varepsilon$ **do** ▷ Standard binary search
    $\gamma \leftarrow (\ell + u)/2$
    $\mathsf{val}_{\mathsf{G}[s=\gamma]} \leftarrow$ SOLVE($\mathsf{G}[s = \gamma]$) ▷ Solution of $\mathsf{G}[s = \gamma]$
    $\gamma' \leftarrow$ ONESTEP($\mathsf{G}, \mathsf{val}_{\mathsf{G}[s=\gamma]}$)($s$) ▷ ONESTEP update of s
    if $\gamma' > \gamma$ **then**
      $[\ell, u] \leftarrow [\gamma, u]$
    else
      $[\ell, u] \leftarrow [\ell, \gamma]$
  return $(\ell + u)/2$

THEOREM 3.1. *Given a game* G *with* k *non-trivial states, a non-trivial state* s*, and an error tolerance of* $\varepsilon > 0$*, the procedure* APPROXSOLVEGUESS *has the following properties:*

- Correctness. *The value* v *returned is an* ε*-approximation of* $\mathsf{val}_{\mathsf{G}}(s)$*, i.e.,* $|v - \mathsf{val}_{\mathsf{G}}(s)| \leq \varepsilon$.
- Complexity. *The procedure calls* SOLVE *at most* $\lceil -\log_2 \varepsilon \rceil$ *times on games with* $(k-1)$ *non-trivial states of the form* $\mathsf{G}[s = \gamma]$*, where* $\gamma = p/q$ *with* $q \leq 2^{\lceil -\log_2 \varepsilon \rceil}$.

Proof. The correctness of APPROXSOLVEGUESS follows from Lemma 3.1. The number of calls follows from standard binary search and the fact that guessing transforms a non-trivial state into a trivial one. ∎

3.4 Approximate to Exact Solver In this section, we show that sufficiently-precise approximation and the exact solution of games are polynomially equivalent tasks. This result is independent of guessing.

Proof overview. The computation of the exact value function given sufficiently precise approximation has three components:

- First, we show that if there is a bound B on value-separation (i.e., values differ by at least $1/B$), then using sufficiently precise approximation the partition of the state space according to the values can be obtained in linear time (see Lemma 3.3).
- Second, we show that given the value partition, the exact value function can be computed in polynomial time (see Lemma 3.4).
- Finally, we present a bound for the value separation (Lemma 3.5).

We formally present the notion of value separation and value partition below.

Value-separation bound B. Consider a game G and a bound B is a *value-separation bound* if for all $s, s' \in S$ either $\mathsf{val}_{\mathsf{G}}(s) = \mathsf{val}_{\mathsf{G}}(s')$ or $|\mathsf{val}_{\mathsf{G}}(s) - \mathsf{val}_{\mathsf{G}}(s')| \geq 1/B$.

Value partition. Consider a game G, a partition $\{X_1, X_2, \ldots\}$ of the states S is called *value partition* if all states in the same set have the same value, and states in different sets have different value, i.e., for all X_i if $x, x' \in X_i$ then $\mathsf{val}_{\mathsf{G}}(x) = \mathsf{val}_{\mathsf{G}}(x')$; and for all $x \in X_i$ and $x' \in X_j$ where $i \neq j$ we have $\mathsf{val}_{\mathsf{G}}(x) \neq \mathsf{val}_{\mathsf{G}}(x')$.

LEMMA 3.3. *Consider a game* G *with a value-separation bound* B*. Consider* $\varepsilon := (5B)^{-1}$*. Then the value partition of* G *can be computed in linear time from the* ε*-approximation of the value function.*

Proof. By solving the approximation of value problem for the game G and error ε, we obtain approximations for each state $\{w_s\}_{s \in S}$ such that, for all $s \in S$, we have $\mathsf{val}_{\mathsf{G}}(s) \in [w_s - \varepsilon, w_s + \varepsilon]$. For $s, s' \in S$, define the relation $s \sim s'$ if $|w_s - w_{s'}| \leq 2\varepsilon$. We show that, by the choice of ε, we have that $s \sim s'$ iff $\mathsf{val}_{\mathsf{G}}(s) = \mathsf{val}_{\mathsf{G}}(s')$.

First, if $s \sim s'$, then

$$|\mathsf{val}_{\mathsf{G}}(s) - \mathsf{val}_{\mathsf{G}}(s')| \leq 2\varepsilon + |w_s - w_{s'}| \leq 4\varepsilon < B^{-1}\,,$$

and so $\mathsf{val}_{\mathsf{G}}(s) = \mathsf{val}_{\mathsf{G}}(s')$.

Conversely, if $\mathsf{val}_{\mathsf{G}}(s) = \mathsf{val}_{\mathsf{G}}(s')$, we have that

$$|w_s - w_{s'}| \leq |w_s - \mathsf{val}_{\mathsf{G}}(s)| + |\mathsf{val}_{\mathsf{G}}(s') - w_{s'}| \leq 2\varepsilon\,,$$

and so, $s \sim s'$. Hence we obtain the value partition in linear time. ∎

LEMMA 3.4. *Consider a game* G *along with its value-partition. Then the value function* $\mathsf{val}_{\mathsf{G}}$ *can be computed in polynomial time.*

Proof. Given the value partition of the game G, consider a player-min's strategy τ as follows. For a state $s \in S_{\min}$, we have $\tau(s) = s'$ such that s and s' belong to the same partition, i.e., $\tau(s) \in \mathsf{OptSucc}(s)$ (recall item (2) from Useful properties of games from Section 2). The strategy τ is an optimal strategy (recall item (3) from Useful properties of games from Section 2). Then, fixing such an optimal strategy τ, an MDP for player-max remains (which is a special case of a stochastic game where every player-min's state has a unique successor). Solving this MDP, which can be done in polynomial time [33, 37, 24], we obtain the exact value function $\mathsf{val}_{\mathsf{G}}$. ∎

We now present a value-separation bound B for stochastic games which proves that the value of two states can not be arbitrarily close. This was proven first for simple stochastic games [12, Lem. 2, p. 208], which is easily extended using the reduction in [42, Sec. 6]. We apply the original proof technique directly to arbitrary rational transition probabilities.

Transitions complexity D**.** Consider a game G with state-space S and rational transition probabilities. We denote the maximum least common multiple of all transition probabilities as D. Formally, for each $s \in S$ we denote its least common multiple of all transition probabilities by

$$D_s \coloneqq \min\{M \in \mathbb{N} : \forall s' \in S \quad M \cdot \Delta(s, s') \in \mathbb{N}\}; \quad \text{and} \quad D \coloneqq \max_{s \in S}\{D_s\}.$$

LEMMA 3.5. *Consider a game* G *with state space* S *and rational transition probabilities with transitions complexity* D*. Then, there is* $q \in \mathbb{N}$ *such that, for all* $s \in S$*, there is* $p_s \in \mathbb{N}$ *such that*

$$\mathsf{val}_{\mathsf{G}}(s) = \frac{p_s}{q}\,,$$

where $0 \leq p \leq q \leq (2D)^{|S|-1}$*, i.e.,* $(2D)^{|S|-1}$ *is a value-separation bound.*

Proof. For a fixed strategy profile π, the value function is the unique solution of $(Id - A)x = b$, where there are subsets $S', S'' \subsetneq S$ such that the matrix $A = (\Delta^{\pi}(s, s'))_{s,s' \in S'}$, the vector $b = \left(\sum_{s' \in S''} \Delta^{\pi}(s, s')\right)_{s \in S'}$, and Id is the identity matrix; see [3, Thm. 10.19, p. 766].

By Cramer's rule [14, p. 656–659], we know an explicit expression for the solution x in terms of the determinant of matrices involving columns of A and b, more specifically, for $0 \leq i \leq n$ we have $x_i = \det B_i / \det(Id - A)$, where B_i is the matrix formed by replacing the i-th column of $(Id - A)$ by the column vector b. The system has a unique solution since $\det(Id - A) \neq 0$. Notice that the determinant of an integer matrix is an integer. Define a diagonal matrix M such that, for all $s, s' \in S$ we have $M_{s,s}\Delta(s, s') \in \mathbb{N}$, the corresponding least common multiple. Then, $\det(M(Id - A)) \in \mathbb{Z}$ and, for all $0 \leq i \leq n$ we have $\det(MB_i) \in \mathbb{Z}$. Thus, the solution x is a rational vector and we only need to bound the size of the denominator. We have

$$\begin{aligned}
|\det M(Id - A)| &= \det M |\det(Id - A)| \\
&= \det M |\det(Id - A^T)| && (A \text{ is square matrix and its transpose is } A^T) \\
&\leq D^{|S'|} \prod_{i=1}^{|S'|} \|(Id - A)_{i,\cdot}\|_1 && (\text{[28, Prop. 5.6.P10, p. 363]}) \\
&\leq D^{|S'|} \prod_{i=1}^{|S'|} \|Id_{i,\cdot}\|_1 + \|A_{i,\cdot}\|_1 && (\text{triangular inequality}) \\
&\leq D^{|S'|} 2^{|S'|} && (\|A_{i,\cdot}\|_1 \leq 1) \\
&\leq (2D)^{|S|-1} && (S' \subsetneq S)\,.
\end{aligned}$$

The desired result for the value-separation bound follows. ∎

REMARK 3.2. *Lemma 3.5 improves previous bounds presented in [27]. Using a similar proof technique within the proof of [27, Thm. 3], they deduce a bound on the denominator q of $|S|^{|S|}d_{max}^{3|S|^2}$, where d_{max} is the maximum denominator of the transition probabilities. Lemma 3.5 is an improvement since $2^{|S|} \leq |S|^{|S|}$ and $D \leq d_{max}^{|S|}$.*

COROLLARY 3.2. *We have a procedure* EXACTDERIVATION *that given a game* G *and an ε-approximation of the value function* $\mathsf{val}_{\mathsf{G}}$*, where $\varepsilon := (2D)^{|S|-1}/5$, returns the exact value function* $\mathsf{val}_{\mathsf{G}}$ *in polynomial time.*

Proof. From Lemma 3.5 the value separation bound is $(2D)^{|S|-1}$. From Lemma 3.3, given the ε-approximation of the value function, the value partition is obtained in linear time. Finally, Lemma 3.4 shows that given the value partition we obtain the value function $\mathsf{val}_{\mathsf{G}}$ in polynomial time. □

3.5 Guessing as Exact Solver In this section, we describe how guessing, which in principle allows only for binary search (i.e. arbitrary approximations), can be used to construct an exact solver for games. We present the algorithm using an oracle, setting the stage for its usage in the context of bounded treewidth games.

Intuitive overview. The idea is to use Corollary 3.2 and Theorem 3.1 recursively as follows. As described in Section 3.4, we can deduce the exact value function of the game from an ε-approximation, provided ε is small enough. To obtain an ε-approximation, we run APPROXSOLVEGUESS in a subset I of non-trivial states. This requires a solver for the exact value function of a simpler game. For this solver, our algorithm can be used as a recursive procedure. Assuming we are given an ε-approximation for all states in the given set I, we can fix their value to this approximation, and then, running the oracle we obtain values for all states outside the set I. Finally, since we have ε-approximations for all states in the game, we can deduce the exact value of the game in all states. This approach is formalized in Algorithm 2.

Reduced game with multiple reductions. We extend the notion of a "reduce game" to allow multiple reductions. For a set of states I and a vector $z = (z_s)_{s\in I}$, we denote $\mathsf{G}[I = z]$ the game with each state $s \in I$ replaced by a trivial state with value z_s. We prove a property of the value function for such games for which we introduce the notion of L_∞ norm for vectors.

Notation for vectors. Given two vectors $(w_s)_{s\in I}$ and $(z_s)_{s\in I}$ we write $\|w - z\|_\infty$ to denote $\max_{s\in I} |w_s - z_s|$.

LEMMA 3.6. *We have $z \mapsto \mathsf{val}_{\mathsf{G}[I=z]}$ is Lipschitz, i.e. for all vectors z' such that $\|z - z'\|_\infty \leq \varepsilon$, we have $\|\mathsf{val}_{\mathsf{G}[I=z]} - \mathsf{val}_{\mathsf{G}[I=z']}\|_\infty \leq \varepsilon$.*

Proof. We consider z' such that $\|z - z'\|_\infty \leq \varepsilon$ and show that for all states s we have $\mathsf{val}_{\mathsf{G}[I=z']}(s) \geq \mathsf{val}_{\mathsf{G}[I=z]}(s) - \varepsilon$, and the other inequality is symmetric.

Fix an optimal strategy σ for player-max in $\mathsf{G}[I = z]$ and consider arbitrary strategy τ for min player. Let π be a strategy profile $\pi = (\sigma, \tau)$. We analyze π and the reachability probability to s_+ in $\mathsf{G}[I = z']$. For a starting state s, the probability to reach s_+ is obtained as the sum of (a) the probability to reach s_+ by never visiting states in I, and (b) the probability to reach s_+ after reaching a state in I. Note that once a state s' in I is reached, the probability to reach s_+ is $z'_{s'}$. Hence we have

$$\begin{aligned}
\mathbb{P}^{\pi}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s, s_+)) &= \mathbb{P}^{\pi}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s, s_+) \cap \mathrm{Safe}(s, I)) + \sum_{s'\in I} \mathbb{P}^{\pi}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s, s')) \cdot z'(s') \\
&\geq \mathbb{P}^{\pi}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s, s_+) \cap \mathrm{Safe}(s, I)) + \sum_{s'\in I} \mathbb{P}^{\pi}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s, s')) \cdot (z(s') - \varepsilon) \\
&\geq \mathbb{P}^{\pi}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s, s_+) \cap \mathrm{Safe}(s, I)) + \sum_{s'\in I} \mathbb{P}^{\pi}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s, s')) \cdot z(s') - \varepsilon \\
&= \mathbb{P}^{\pi}_{\mathsf{G}[I=z]}(\mathrm{Reach}(s, s_+) \cap \mathrm{Safe}(s, I)) + \mathbb{P}^{\pi}_{\mathsf{G}[I=z]}(\mathrm{Reach}(s, s_+) \cap \mathrm{Reach}(s, I)) - \varepsilon \\
&= \mathbb{P}^{\pi}_{\mathsf{G}[I=z]}(\mathrm{Reach}(s, s_+)) - \varepsilon \\
&\geq \mathsf{val}_{\mathsf{G}[I=z]}(s) - \varepsilon
\end{aligned}$$

The first inequality follows since $\|z - z'\|_\infty \leq \varepsilon$. The second ineqaulity follows since $\sum_{s'\in I} \mathbb{P}^{\pi}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s, s')) \cdot \varepsilon = \mathbb{P}^{\pi}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s, I)) \cdot \varepsilon \leq \varepsilon$, as the probability to reach a set is at most 1. The following two equalities are obtained from the description of decomposing the reachability probability above. The final inequality follows from optimality of σ in $\mathsf{G}[I = z]$.

Hence given σ and for state s we have $\min_{\tau\in\Gamma_{\mathsf{G}[I=z']}} \mathbb{P}^{\sigma,\tau}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s,s_+)) \geq \mathsf{val}_{\mathsf{G}[I=z]}(s) - \varepsilon$, which gives

$$\max_{\sigma\in\Sigma_{\mathsf{G}[I=z']}} \min_{\tau\in\Gamma_{\mathsf{G}[I=z']}} \mathbb{P}^{\sigma,\tau}_{\mathsf{G}[I=z']}(\mathrm{Reach}(s,s_+)) \geq \mathsf{val}_{\mathsf{G}[I=z]}(s) - \varepsilon$$

and finally $\mathsf{val}_{\mathsf{G}[I=z']} \geq \mathsf{val}_{\mathsf{G}[I=z]} - \varepsilon$ and that concludes the proof. ∎

Algorithm 2 Solve by guessing

Input: Game G, set of states I, oracle for the value problem SOLVE
Output: Value function $\mathsf{val}_{\mathsf{G}}$

1: **procedure** SOLVEGUESS(G, I, SOLVE)
2: $D \leftarrow$ Transitions complexity of G
3: $\varepsilon \leftarrow (2D)^{|S|-1}/5$
4: **for** $s \in I$ **do**
5: ORACLE$(\cdot) \leftarrow$ SOLVEGUESS$(\cdot, I \setminus \{s\},$ SOLVE$)$ ▷ Define ORACLE function
6: $z_s \leftarrow$ APPROXSOLVEGUESS$(\mathsf{G}, s, \varepsilon,$ ORACLE$)$ ▷ Approximate value of state s
7: $w \leftarrow$ SOLVE$(\mathsf{G}[I = z])$ ▷ Solve reduced game
8: $\mathsf{val}_{\mathsf{G}} \leftarrow$ EXACTDERIVATION(G, w)
9: **return** $\mathsf{val}_{\mathsf{G}}$

Informal description of Algorithm 2: Procedure SOLVEGUESS. In Line 2 and Line 3 the transition complexity and the value separation bound is computed. In Line 4, we iterate over all states in I, searching for an ε-approximation. In Line 5, we create a function ORACLE that takes a game and returns a solution, since this is required as input to the function APPROXSOLVEGUESS. The oracle is obtained as recursive use of the procedure itself on simpler games (with less non-trivial states). In Line 6, we compute approximation of the value function using APPROXSOLVEGUESS with the ORACLE as input. This will terminate because SOLVEGUESS will be called with a game that has one less non-trivial state. In Line 7, the oracle is used to obtain ε-approximations for all states from the game $\mathsf{G}[I = z]$ where all states in I are trivial. Finally, in Line 8, we apply Corollary 3.2 to compute the exact value function from the previously computed ε-approximations.

Just as in Theorem 3.1, we analyse the complexity of Algorithm 2 in terms of the number of calls to the oracle.

THEOREM 3.2. *Given a game* G*, a set of states* I*, an oracle for the value problem* SOLVE*, and transitions complexity* D*, the procedure* SOLVEGUESS *has the following properties:*

- Correctness. *The value function* $\mathsf{val}_{\mathsf{G}}$ *returned is the value function of* G.
- Complexity. *The procedure calls* SOLVE *at most* $\mathcal{O}\left((|I||S|^2 \log D)^{|I|}\right)$ *times on games where all states in* I *are trivial.*

Proof. We present the correctness argument and then the complexity analysis.

Correctness. Notice that in Line 6, for all states $s \in I$ we have z_s is an ε-approximation of $\mathsf{val}_{\mathsf{G}}(s)$, by Theorem 3.1. Denote $v \coloneqq (\mathsf{val}_{\mathsf{G}}(s))_{s\in I}$, and note that $\mathsf{val}_{\mathsf{G}[I=v]} = \mathsf{val}_{\mathsf{G}}$. Then, since $z \mapsto \mathsf{val}_{\mathsf{G}[I=z]}$ is Lipschitz from Lemma 3.6 and $\|z - v\|_\infty \leq \varepsilon$, we have that $w = \mathsf{val}_{\mathsf{G}[I=z]}$ is an ε-approximation of $\mathsf{val}_{\mathsf{G}}$. Indeed, $\|w - \mathsf{val}_{\mathsf{G}}\|_\infty = \|\mathsf{val}_{\mathsf{G}[I=z]} - \mathsf{val}_{\mathsf{G}[I=v]}\|_\infty \leq \|z - v\|_\infty \leq \varepsilon$. That means w on Line 7 is an ε-approximation of the value function. Since $\varepsilon = (2D)^{|S|-1}/5$, the procedure EXACTDERIVATION computes the exact value function (see Corollary 3.2).

Compexity. Let us now count the number of times the oracle is called. During the execution of Algorithm 2, for each $s \in I$, a recursion tree is formed where the root is G, its children have one guessed state, their children have one more guessed state, and so on. Denote $\mathsf{G}^{(i)}$ a game where i states have been guessed and $D(\mathsf{G}^{(i)})$ its transitions complexity. Notice that Algorithm 2 only calls the oracle SOLVE when all states in I have been guessed. Therefore, we only need to keep track of two parameters: transitions complexity and the number of non-trivial states in I.

- On the transitions complexity, note that guessing a state only affects its own transitions. Therefore, for any $i \in \{0, \ldots, |I|\}$, the transitions complexity of $\mathsf{G}^{(i)}$ is not larger than $\max\{D, 2^{\lceil -\log_2(\varepsilon)\rceil}\} = \mathcal{O}(D^{|S|})$.
- On the number of non-trivial states in I, notice that all guessed states must be in I. Therefore, for all $i \in \{0, \ldots, |I|\}$, $\mathsf{G}^{(i)}$ contains $(|I| - i)$ non-trivial states in I.

We deduce the number of times time SOLVE is called during an excecution of Algorithm 2 as follows.

$$
\begin{aligned}
\mathcal{O}(\textsc{SolveGuess}) &\leq |I| \cdot \mathcal{O}\left(\textsc{ApproxSolveGuess}(\mathsf{G}, \varepsilon, \textsc{SolveGuess})\right) + 1 && \text{(Line 4)}\\
&\leq |I| \log(\varepsilon) \cdot \mathcal{O}\left(\textsc{SolveGuess}(\mathsf{G}^{(1)})\right) && \text{(Theorem 3.1)}\\
&= |I||S| \log\left(D\left(\mathsf{G}^{(0)}\right)\right) \cdot \mathcal{O}\left(\textsc{SolveGuess}(\mathsf{G}^{(1)})\right) && \left(\varepsilon \in \mathcal{O}\left(D^{|S|}\right)\right)\\
&\leq |I||S|^2 \log(D) \cdot \mathcal{O}\left(\textsc{SolveGuess}(\mathsf{G}^{(1)})\right) && \left(D(G^{(i)}) \in \mathcal{O}\left(D^{|S|}\right)\right)\\
&\leq |I|(|I|-1)\left(|S|^2 \log(D)\right)^2 \cdot \mathcal{O}\left(\textsc{SolveGuess}(\mathsf{G}^{(2)})\right) && \text{(Theorem 3.1)}\\
&\leq \dots\\
&\leq \mathcal{O}\left(|I|! \cdot \left(|S|^2 \log(D)\right)^{|I|}\right)\\
&\leq \mathcal{O}\left(\left(|I||S|^2 \log(D)\right)^{|I|}\right) && |I|! \leq |I|^{|I|}.
\end{aligned}
$$

The desired result follows. ∎

COROLLARY 3.3. *Consider a variant of Algorithm 2 for a game* G *where we fix* $I = S$ *(and the oracle is now irrelevant). The value function is computed in time* $\mathcal{O}((|S|^3 \log D)^{|S|})$*, where* S *is the set of states and* D *is the transitions complexity.*

4 Faster Algorithm for Games with Bounded Treewidth

In this section, we first observe how to decompose games. Then we introduce the notion of bounded treewidth and balanced separators. Finally, we show how to use Algorithm 2 on games where the underlying graph has bounded treewidth to get a faster algorithm.

4.1 Decomposing Games

Recall that the reachability value of a state s can be deduced from the values of its successors. Therefore, if we can separate a game into two sets of states (X and Y) where the values of states in X are independent of those in Y, then we can first solve X and then, using the obtained values, solve Y. This is formalized in the following lemma.

LEMMA 4.1. *Let* G *be a game. If there exists* X, Y*, a partition of* S *such that, for all strategy profiles* π*, states* $x \in X$*, and states* $y \in Y$*, we have that* $\mathbb{P}^{\pi}_{\mathsf{G}}(\mathrm{Reach}(x, y)) = 0$*, then the coordinates of* $\mathsf{val}_{\mathsf{G}}$ *corresponding to states in* X *can be solved independently of those in* Y*.*

Proof. Since states in X cannot reach states in Y, their values are independent of Y. Thus, we first solve the sub-game restricted to set X. Now, with $\mathsf{val}_{\mathsf{G}}(x)$ for all $x \in X$, we change X to trivial states with corresponding values. Then we solve a game with non-trivial states only in Y. ∎

Guessing and dependency reduction. Generally, games do not have a structure that would allow to exploit the above lemma. However, recall that Algorithm 2 modifies the game graph. It makes some states trivial, which disconnects them from all their successors. In other words, guessing breaks dependencies between states. Suppose that a game consists of sets of states X and Y such that there are only a few transitions from X to Y. By guessing the value of states from X with a transition to Y, we remove the dependency of X on Y and we can solve game restricted on X independently. We show that bounded treewidth allows us to exploit this idea.

4.2 Treewidth

In this section, we briefly recall the notion of treewidth and how to find a balanced separator, see [18, Sec. 12.4, p. 355] for an overview. Intuitively, the treewidth of a graph describes how "far" the graph is from being a tree.

Tree decomposition of graphs and treewidth. Given a graph $G = (S, E)$ with vertex set (or state space) S and edge set E, a *tree decomposition* $\mathcal{T} = (T, E_T)$ of G is a tree, where we refer to nodes of the tree as bags. Every bag Bg represent a set of states of G. The tree $\mathcal{T}$ satisfies the following properties: (i) $\bigcup_{Bg \in T} Bg = S$ (i.e., the union of states in the bags cover the state space); (ii) with $E_{Bg} \coloneqq \{(u, v) \in E \mid u, v \in Bg\}$, we have $\bigcup_{Bg \in T} E_{Bg} = E$ (i.e., edges of E are covered by edges of vertices in bags); and (iii) for all $X, Y, Z \in T$, if Y is on the unique path from X to Z, then $X \cap Z \subseteq Y$. The *width* of a tree decomposition is the maximal size of a bag minus one, i.e. $\max_{Bg \in T} |Bg| - 1$. For a graph G, the *treewidth* of G is the minimum width over all tree decompositions of G.

Balanced separators. We consider the notion of *balanced separators* and follow the definition from [15, Sec. 7.6.1]. For a graph $G = (S, E)$ we call a set of states I a *balanced separator* if in the sub-graph induced after removing I, i.e., in $(S' = S \setminus I, E' = E \cap (S' \times S'))$, there is a partition $C_1, C_2, \ldots$ of S' such that every partition set C_i has size at most $\lfloor |S|/2 \rfloor$ and there are no edges across the partition sets. For computation of balanced separators, there exists a procedure SEPARATOR, that takes as input a graph and a tree decomposition of width t and in time $\mathcal{O}(t|S|)$ returns a balanced separator I of size at most t along with the partition $C_1, C_2, \ldots$ (see [15, Sec. 7.6.1]).

REMARK 4.1. *Observe that treewidth and balanced separator is more restrictive than what is needed: removing the balanced separator ensures that there is no path from X to Y as well as from Y to X. In contrast, Lemma 4.1 only requires that path in one direction (from X to Y) is not there, rather than in both directions. In other words, instead of cuts in the graph provided by treewidth, our approach only requires* directed *cuts.*

4.3 Improved Algorithm We combine the insights of the previous sections to obtain a faster algorithm for games with bounded treewidth.

Intuitive overview. We present our approach in Algorithm 3 which proceeds as follows. Consider a game G with the tree decomposition $\mathcal{T}$ of width t of the associated graph $\widehat{G}$ of the game, it identifies a balanced separator I by the procedure SEPARATOR. Then, it applies SOLVEGUESS on all non-trivial states in I, making all states from I trivial. This splits the game graph of G into partitions each at most half the size of the original game which are solved recursively.

Basic procedures for Algorithm 3. We present three procedures that the algorithm requires and all these procedures have linear running time.

- DECOMPOSE: This function takes a game as an argument and returns sets of (smaller) games. The function removes the set of states X which consists of s_+, s_- and all trivial states. Then the function partitions $S \setminus X$ to sets $\{S_1, S_2, \ldots\}$ such that for every $s \in S_i$ holds $E(s) \subseteq S_i \cup X$. The partitions are achieved as follows. let $Z := X$ and we consider the graph induced by $S \setminus Z$, we pick a state s and compute the set of states that are either reachable from s or can reach s, which defines the partition S_1; we then set $Z := Z \cup S_1$ and continue as above to find S_2 and so on. This is achieved by simple graph reachability (e.g., DFS) and is linear time. Finally, the set of games $\{C_1, C_2, \ldots\}$ is returned where C_i is a game restricted on states $S_i \cup X$.

- RESTRICT: This function takes as an argument a tree decomposition $\mathcal{T}$ of a graph $\widehat{G} = (S, E)$ and set $C \subseteq S$ and restricts $\mathcal{T}$ to the graph $(C, (C \times C) \cap E)$. The function just removes $S \setminus C$ from all bags of $\mathcal{T}$, and returns the resulting tree decomposition.

- COMPOSE: This function takes as argument two vectors v_1, v_2 indexed by states from S and returns a vector v such that $v(s) = \max(v_1(s), v_2(s))$.

Algorithm 3 Quasipolynomial algorithm for bounded treewidth

Input: Game G, and tree decomposition $\mathcal{T}$
Output: $\mathsf{val}_{\mathsf{G}}$

```
1:  procedure RecursiveSolve(G, T)
2:      if G contains only trivial states then
3:          return OneStep(G, {val_G(s_+) ↦ 1, val_G(s_-) ↦ 0})      ▷ OneStep update of trivial states
4:      C ← Decompose(G)
5:      v ← 0                                                        ▷ Zero vector to store the solution
6:      for C ∈ C do
7:          T_C ← Restrict(T, C)
8:          I ← Separator(T_C)
9:          Oracle(·) ← RecursiveSolve(·, T_C)
10:         val_C ← SolveGuess(C, I, Oracle)
11:         v ← Compose(v, val_C)
12:     return v
```

THEOREM 4.1. *Given a game* G *with state space* S *and transitions complexity* D *and a tree decomposition* $\mathcal{T}$ *of the game graph of width* t, RECURSIVESOLVE$(\mathsf{G}, \mathcal{T})$ *computes* $\mathsf{val}_{\mathsf{G}}$ *in time* $\mathcal{O}((t|S|^2 \log D)^{t \log |S|})$.

Proof. We present the correctness argument and complexity analysis.

Correctness. First note that the partitions obtained DECOMPOSE(G) satisfy the property of Lemma 4.1. Hence from Lemma 4.1, the procedure RECURSIVESOLVE can solve every set returned from DECOMPOSE(G) independently. The correctness of the solution for every set follows from the correctness of SOLVEGUESS (Theorem 3.2). Also, note that when the values are composed using COMPOSE the states in X are common where the values are fixed (as they are trivial states) and the other parts of the games to compute values are disjoint. Hence the correctness follows.

Complexity analysis. We analyse the complexity of the recursive procedure.

- *Base case.* If there are only trivial states in G, then RECURSIVESOLVE returns the solution in time linear in the number of states.
- *Recursive case.* If for G, the SEPARATOR returns set I, then from Theorem 3.2, we know that SOLVEGUESS computes $\mathsf{val}_{\mathsf{G}}$ and the number of calls to ORACLE is bounded by $\mathcal{O}((|I||S|^2 \log D)^{|I|})$ times. Moreover, in the calls to ORACLE, the states in I are trivial.

Observe that after making separator trivial, all subgames contain at most half of the non-trivial states. This implies that the depth of the recursion is at most $\log |S|$. Hence, the overall time complexity of the algorithm is $\mathcal{O}((t|S|^2 \log D)^{t \log |S|})$. □

Space requirement. To solve one step of the recursion, we need to store upper and lower bounds for states and transition probabilities which are polynomial at any recursion depth. The maximal recursion depth is $t \log |S|$. Hence the required space is polynomial.

We note that for a graph G with n states and treewidth t, an optimal tree-decomposition can be constructed in time $O(n^{t+2})$ (see [2]). Thus invoking Theorem 4.1 after computing an optimal tree-decomposition we obtain the following corollary.

COROLLARY 4.1. *Given a game* G *with* n *states, treewidth* t*, and transition complexity* D*, the value function* $\mathsf{val}_{\mathsf{G}}$ *can be computed in time* $\mathcal{O}((tn^2 \log D)^{t \log n})$.

5 Discussion

We discuss the implication of our result, its connection to mean-payoff games, and future directions.

Basic implications. The basic implications of our result are as follows.

- *Quasi-polynomial for small treewidth.* For stochastic games G with n states where the treewidth t is constant or poly-logarithmic in n, our deterministic algorithm requires quasi-polyonomial time.
- *Sub-exponential for large class.* For stochastic games G with n states where the treewidth t is $\mathcal{O}(n^{1-\varepsilon})$ for $\varepsilon > 0$, our deterministic algorithm requires sub-exponential time.

Deterministic mean-payoff games. Mean-payoff games are two-player deterministic games, which have no probabilistic states. Instead every edge is assigned an integer reward/cost in $\{-W, \ldots, W\}$. For a strategy profile, the infinite path/walk, consists of a finite prefix, followed by a cycle C repeated infinitely often, and the payoff is the sum of the weights of C divided by the length C (i.e., mean weight of the cycle). The goal of player-max is to maximize the payoff against adversarial player-min. A reduction from mean-payoff games to turn-based stochastic games is presented in [42]. We argue that the reduction increases the treewidth by at most 2. The main idea of the reduction is as follows (see [42] for details). We add two states s_+ and s_-, and for every edge (u, v) with weight w, we add a new probabilistic state $x_{u,v}$ with

$$\Delta(x_{u,v}, y) = \begin{cases} \lambda & y = v; \\ (1-\lambda)\frac{W+w}{2W} & y = s_+; \\ (1-\lambda)\frac{W-w}{2W} & y = s_-; \end{cases}$$

where $\lambda = 1 - \frac{1}{4n^3W}$. Observe that the common divisor of all probabilities is $8n^3W^2$, i.e., for the obtained stochastic game the transitions complexity D is $\mathcal{O}(n^3W^2)$. Since subdividing edges does not increase the treewidth and states s_+ and s_- can

be added to every bag of the tree decomposition, it follows that the treewidth increases by at most 2. Thus all of our results for turn-based stochastic games apply to mean-payoff games with treewidth increased by 2 and $\mathcal{O}(\log D) = \mathcal{O}(\log nW)$.

Future works. There are many interesting future directions towards the major open question of polynomial-time algorithm for stochastic games. Two immediate open questions are whether (i) stochastic games with constant treewidth can be solved in polynomial time; and (ii) stochastic games can be solved in deterministic sub-exponential time.

Acknowledgements

This research was partially supported by the ERC CoG 863818 (ForM-SMArt) grant.

References

[1] D. Andersson and P. Miltersen, *The complexity of solving stochastic games on graphs*, Algorithms and Computation, Springer Berlin Heidelberg, 2009.
[2] S. Arnborg, D. G. Corneil, and A. Proskurowski, *Complexity of finding embeddings in a k-tree*, SIAM JOURNAL OF DISCRETE MATHEMATICS, 8(2) (1987), pp. 277–284.
[3] C. Baier and J. Katoen, *Principles of Model Checking*. MIT Press, 2008.
[4] H. L. Bodlaender, *Dynamic programming on graphs with bounded treewidth*, Automata, Languages and Programming, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 317 (1988), pp. 105–118.
[5] L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J. Raskin, *Faster algorithms for mean-payoff games*, Formal Methods Syst. Des., 38(2) (2011), pp. 97–118.
[6] C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan, *Deciding parity games in quasi-polynomial time*, SIAM J. Comput., 51(2) (2022), pp. 17–152.
[7] K. Chatterjee and N. Fijalkow, *A reduction from parity games to simple stochastic games*, GandALF 2011, EPTCS, 54 (2011), pp. 74–86.
[8] K. Chatterjee, A. K. Goharshady, P. Goyal, R. Ibsen-Jensen, and A. Pavlogiannis, *Faster algorithms for dynamic algebraic queries in basic RSMs with constant treewidth*, ACM Trans. Program. Lang. Syst., 41(4) (2019), pp. 23:1–23:46.
[9] K. Chatterjee and T. A. Henzinger, *Value iteration*, 25 Years of Model Checking - History, Achievements, Perspectives, Lecture Notes in Computer Science, Springer, 5000 (2008), pp. 107–138.
[10] K. Chatterjee, R. Ibsen-Jensen, and A. Pavlogiannis, *Faster algorithms for quantitative verification in bounded treewidth graphs*, Formal Methods Syst. Des., 57(3) (2021), pp. 401–428.
[11] A. Condon, *On Algorithms for Simple Stochastic Games*, Advances in computational complexity theory, 1990.
[12] A. Condon, *The complexity of stochastic games*, Information and Computation, 96(2) (1992), pp. 203–224.
[13] B. Courcelle, *Graph rewriting: An algebraic and logic approach*, Handbook of Theoretical Computer Science, Elsevier and MIT Press, Volume B: Formal Models and Semantics (1990), pp. 193–242.
[14] G. Cramer, *Introduction à l'analyse des lignes courbes algébriques*, 1750.
[15] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer Publishing Company, Incorporated, 1st edition, 2015.
[16] L. de Alfaro and R. Majumdar, *Quantitative solution of omega-regular games*, Journal of Computer and System Sciences, 68(2) (2004), pp. 374–397.
[17] X. B. de Montjoye, *A recursive algorithm for solving simple stochastic games*, arXiv preprint arXiv:2110.01030, 2021.
[18] R. Diestel, *Graph Theory*, Springer Berlin Heidelberg, New York, NY, 2017.
[19] D. Dorfman, H. Kaplan, and U. Zwick, *A faster deterministic exponential time algorithm for energy games and mean payoff games*, ICALP 2019, LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 132 (2019), pp. 114:1–114:14.
[20] A. Ehrenfeucht and J. Mycielski, *Positional strategies for mean payoff games*, Int. Journal of Game Theory, 8(2) (1979), pp. 109–113.
[21] E.A. Emerson and C.S. Jutla, *Tree automata, mu-calculus and determinacy*, FOCS, (1991), pp. 368–377.
[22] H. Everett, *Recursive games*, Contributions to the Theory of Games III, Annals of Mathematical Studies, 39 (1957), pp. 47–78.
[23] J. Fearnley and S. Schewe, *Time and parallelizability results for parity games with bounded treewidth*, ICALP 2012, Lecture Notes in Computer Science, Springer, 7392 (2012), pp. 189–200.
[24] J. Filar and K. Vrieze, *Competitive Markov Decision Processes*, Springer-Verlag, 1997.
[25] H. Gimbert and F. Horn, *Solving Simple Stochastic Games with Few Random Vertices*, Logical Methods in Computer Science, 2009.
[26] V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan, *Cyclic games and an algorithm to find minimax cycle means in directed graphs*, USSR Comput. Math. Math. Phys., 28(5) (1990), pp. 85–91.
[27] S. Haddad and B. Monmege, *Interval iteration algorithm for MDPs and IMDPs*, Theoretical Computer Science, 735 (2018), pp. 111–131.
[28] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 2013.

[29] R. Ibsen-Jensen and P. Miltersen, *Solving simple stochastic games with few coin toss positions*, ESA 2012, Lecture Notes in Computer Science, Springer, 7501 (2012), pp. 636–647.
[30] M. Jurdzinski, *Deciding the winner in parity games is in UP ∩ co-UP*, Information Processing Letters, 68(3) (1998), pp. 119–124.
[31] M. Jurdzinski, *Small progress measures for solving parity games*, STACS 2000, Springer, (2000), pp. 290–301.
[32] M. Jurdzinski, M. Paterson, and U. Zwick, *A deterministic subexponential algorithm for solving parity games*, SIAM J. Comput., 38(4) (2008), pp. 1519–1532.
[33] L. G. Khachiyan, *Polynomial algorithms in linear programming*, USSR Computational Mathematics and Mathematical Physics, 20(1) (1980), pp. 53–72.
[34] W. Ludwig, *A subexponential randomized algorithm for the simple stochastic game problem*, Information and Computation, 117 (1995), pp. 151–155.
[35] M. Melekopoglou and A. Condon, *On the Complexity of the Policy Improvement Algorithm for Markov Decision Processes*, ORSA Journal on Computing, 6(2) (1994), pp. 188–192.
[36] J. Obdrzálek, *Fast mu-calculus model checking when tree-width is bounded*, CAV 2003, Lecture Notes in Computer Science, Springer, 2725 (2003), pp. 80–92.
[37] M. L. Puterman, *Markov Decision Processes*, John Wiley and Sons, 1994.
[38] S. Schewe, *Solving parity games in big steps*, FSTTCS 2007, 84 (2007), pp. 449–460.
[39] L. S. Shapley, *Stochastic Games*, 39(10) (1953), pp. 1095–1100.
[40] M. Thorup, *All Structured Programs Have Small Tree Width and Good Register Allocation*, Information and Computation, 142(2) (1998), pp. 159–181.
[41] W. Zielonka, *Infinite games on finitely coloured graphs with applications to automata on infinite trees*, Theoretical Computer Science, 200(1-2) (1998), pp. 135–183.
[42] U. Zwick and M. Paterson, *The complexity of mean payoff games on graphs*, Theoretical Computer Science, 158(1) (1996), pp. 343–359.