Accelerated Approximation for Stochastic Reachability Games[*]

Rafał Somla[†]

Abstract

In this paper new algorithms for finding optimal values and strategies in turn-based stochastic games with reachability objectives are presented, whose special case are the simple stochastic games considered elsewhere [4, 11]. The general idea of these algorithms is to accelerate the successive approximation scheme for solving stochastic games [13] in which node values are updated in each iteration so that they converge to the optimal values of the game. This scheme is extended with a pair of positional strategies which are updated to remain greedy with respect to the current approximation. This way optimal strategies can be discovered before the current values get close to the optimal ones. The approximation process is accelerated, by predicting an approximate result of several updates of the current valuation and jumping directly to the predicted values.

New algorithms are based on three different acceleration techniques: iterative squaring, linear extrapolation, and linear programming; with different difficulty of performing single iteration and different acceleration level achieved by each of them. For each of these algorithms the complexity of a single iteration is polynomial.

It is shown that accelerated algorithms will never perform worse than the basic, non-accelerated one and exponential upper bounds on the number of iterations required to solve a game in the worst case is given. It is also proven that new algorithms increase the frequency with which the greedy strategies are updated. The more often strategies are updated, the higher chances that the algorithm will terminate early. It is proven that the algorithm based on linear programming updates the greedy strategies in every iteration, which makes it similar to the strategy improvement method, where also a new strategy is found in each iteration and this also at the cost of solving linear constraint problems

[*]Extended version of paper *New algorithms for solving simple stochastic games* publised in *Proceedings of the Workshop on Games in Design and Verification (GDV 2004)*, volume 119 of *Electronic Notes in Theoretical Computer Science*, pages 51–65. Elsevier.

[†]Department of Information Technology, Uppsala University.

Paper is concluded with presentation of results of experiments in which new algorithms were run on a sample of randomly generated games. It could be observed that the proposed acceleration techniques reduce the number of iterations of the basic algorithm by an order of magnitude and that they substantially increase frequency with which the greedy strategies are updated. The algorithms based on linear programming and linear extrapolation displayed similar efficiency as the ones based on strategy improvement. This makes the algorithm based on linear extrapolation especially attractive because it uses much simpler computations than linear constraint solving.

Contents

1 Introduction

Stochastic games of the type studied here are now well understood. The existence of optimal values and strategies have been proven for games with a wide range of winning conditions. However, not much is known about how to efficiently compute these optimal values and strategies.

It seems that introducing probabilistic moves in a game greatly increases the difficulty of solving such a game. In the case of deterministic games, there are polynomial time algorithms available when the winning conditions are simple. For instance deterministic reachability/safety games can be easily solved in time linear in the size of the game board. Also for parity winning conditions with fixed number of priorities the existing algorithms are polynomial. For general parity winning conditions, advanced combinatorial optimization methods have been proposed [1, 9] which lead to randomized, and subsequently also deterministic, sub-exponential decision procedures.

However, even for the very simple reachability winning conditions, it is unknown whether the corresponding stochastic games can be solved in a polynomial time. Known reductions show that these games are already as difficult as deterministic games with general parity winning conditions or with limit-average payoffs. This also means that the important problem of verifying properties expressed in the modal μ-Calculus can be reduced to the problem of determining optimal values of a stochastic reachability game. Whether any of these problems has a polynomial time decision procedure is a long-standing open question which could not be answered despite many years of intensive research.

In this paper stochastic games with reachability winning conditions are investigated, as they already form a challenging subject of study from the computational perspective. A special case of such games are the simple stochastic games studied by Condon [4, 5] and Ludwig [11]. As mentioned above, any new techniques developed for this type of games can be also applied to mean-payoff or parity games, shedding new light on the difficult and intriguing question of their exact complexity.

The main focus in this paper is placed on the practical question of how to compute the optimal values and strategies for a game in an efficient way. Starting from a very simple successive approximation method of determining the optimal values of a game, three acceleration techniques are proposed which improve its convergence rate. These acceleration techniques allow one to trade the complexity of performing a single iteration for the number of iterations which lead to the solution.

After defining stochastic reachability games, their optimal values and strategies, Shapley's proof of existence of optimal values [13] is recalled, adapted to our setting and extended to arbitrary games (Shapley's original argument uses assumption that games stop with probability one). Embedded in the proof of Shapley is a simple algorithm for determining optimal values by a successive approximation. A termination criteria is added to it, based on stable positional strategies and it is shown that resulting algorithm must find optimal strategies and values after at most $\lg(D) \cdot N^2 \cdot D^N$ iterations, where N is the number of game positions and $\lg(D)$ is roughly the number of bits needed to represent probabilities of random moves.

Next, three different techniques for accelerating the basic approximation scheme are presentd. They try to increase the progress made in a single iteration of the algorithm using three different heuristics. General conditions under which such acceleration schemas are correct are given, which guarantee that optimal values of the game are found despite the fact that accelerated algorithms depart from the sequence of valuations computed by the non-accelerated version. The complexity of a single iteration of each accelerated algorithm is analyzed and proved to be polynomial.

Paper is concluded with presentation of results of experiments in which accelerated algorithms are run on random games and the number of iterations needed to solve the games is measured. These experimental results confirm that the heuristics used to accelerate the basic approximation scheme substantially improve its convergence rate. New algorithms were also compared with ones based on strategy improvement. The latter are an alternative approach to solving stochastic games which proves to be very efficient in practice although computational costs of a single iteration are high. It was observed that acceleration based on linear optimization and linear extrapolation gives similar efficiency as that of the strategy improvement algorithms. In view of these experiments, the algorithm based on linear extrapolation seems to provide best compromise between the complexity of a single iteration and the number of iterations needed to solve an average game.

2 Preliminaries

In this paper we consider stochastic games played between two players on a finite directed graph with reachability winning conditions. The graph represents a game board on which two players called **max** and **min** move a token along the edges. The goal of player **max** is to reach a designated target position, while player **min** tries to spoil these attempts. Certain positions in the game are random. When the token reaches such a position it is moved randomly according to a given probability distribution over its successors. Positions which are not random are controlled by one of the players who chooses where to move the token.

Definition 1 (game board) *A* reachability game board *is a tuple*

$$\langle V, V_{\max}, V_{\min}, E, p, t\rangle$$

where $\langle V, E\rangle$ is a directed graph (called the game graph) whose nodes are game positions. We write $x \to y$ if $(x,y) \in E$.

Disjoint sets $V_{\max}, V_{\min} \subseteq V$, are sets of positions controlled by player **max** *and* **min**, *respectively. The set of random positions is defined as $V_{\mathrm{rnd}} = V \setminus (V_{\max} \cup V_{\min})$. For $(x,y) \in E$ with $x \in V_{\mathrm{rnd}}$, $p(x,y) > 0$ is the probability of moving the token from x to y. We assume $\sum_{x\to y} p(x,y) = 1$ for $x \in V_{\mathrm{rnd}}$. Sink $t \in V$ is the target position.*

During a play on a game board G, a possibly infinite path through the game graph is formed by the moving token.

Definition 2 (plays) *Let $G = \langle V, V_{\max}, V_{\min}, E, p, t\rangle$ be a game board. A* play *on G is a sequence $x_0, x_1, \ldots$ of game positions such that $(x_i, x_{i+1}) \in E$ for all i. We require this sequence to be maximal in the sense that it is finite only if its last position x_n is terminal, i.e. has no successors in G.*

To play the game one also needs to specify the starting position. Thus given game board G different games can be played starting from different positions $x \in V$. We will write $G(x)$ to denote the game starting at x.

A *strategy* for a player is a plan determining the choices he makes during any possible play of the game. In general, this choice can depend on the *history* of the play, that is on the sequence of game positions visited so far.

Definition 3 (strategy) *Let G be a game board. A* strategy *for player $p \in \{\mathbf{max}, \mathbf{min}\}$ in G is a function $\sigma : V_p \times V^* \to V$ such that for any $x \in V_p$ and any history $h \in V^*$ if $y = \sigma(x;h)$ then $(x,y) \in E$.*

Note that we require from a strategy slightly more than necessary, since it must prescribe a move for any sequence of game positions regardless of whether it is a valid history of a play or not. However, it is easy to extend any strategy to such a total function by adding arbitrary choices for sequences which are not relevant.
A play in which a player uses given strategy is said to conform to that strategy.

Definition 4 *Let G be a reachability game board. A play $x_0, x_1, \ldots$ on G* conforms *to strategy $\sigma : V_p \times V^* \to V$ for player $p \in \{\mathbf{max}, \mathbf{min}\}$ iff*

$$x_{i+1} = \sigma(x_i; x_0 \cdots x_{i-1})$$

for any $x_i \in V_p$.

A special case of a strategy is when the choice does not depend on the history of a play but only on the current position. Such strategies are called positional or memoryless.

Definition 5 (positional strategy) *A strategy $\sigma : V_p \times V^* \to V$ is* positional *if $\sigma(x,h) = \sigma(x,h')$ for all $x \in V_p$ and any histories $h, h' \in V^*$.*

A positional strategy for player p can be considered as a function $\sigma : V_p \to V$.

Since there are random positions in a game, players do not have the full control over the course of a play. With their choices they can only try to maximize the probability of winning a random play.

Let us fix starting position $x \in V$ on a game board G and strategies (not necessarily positional) σ and τ for both players. They determine a unique probability measure $\mathrm{Pr}^x_{\sigma,\tau}$ on plays from x conforming to σ and τ, which agrees with the edge probabilities of the game board. In particular, the probability $\mathrm{Pr}^x_{\sigma,\tau}(p)$ of a finite play p from x conforming to strategies σ and τ equals the product of edge probabilities along this play.

Definition 6 (winning probability) *Let $G(x)$ be a stochastic reachability game played on game board G. The value $v_{\sigma,\tau}(x)$ for strategies σ and τ in game $G(x)$ is the probability that a play conforming to σ and τ reaches target t under the probability measure $\mathrm{Pr}^x_{\sigma,\tau}$.*

Often, for a given game board we will consider $v_{\sigma,\tau}$ as a valuation of game positions, that is, a function $v_{\sigma,\tau} : V \to [0,1]$.

Player's **max** goal is to reach the target. Hence $v_{\sigma,\tau}(x)$ is the probability that **max** wins a play starting at x and conforming to σ and τ. At the same time $1 - v_{\sigma,\tau}(x)$ is the probability that player **min** wins such a play. Thus maximizng probability of winning a game by **min** is the same as minimizing the winning probability of player **max** which gives names to the players.

An optimal value for a player in game $G(x)$ is the maximal probability of winning the game by that player given that he makes optimal choices and regardless of how well the opponent is defending.

Definition 7 (optimal values) *The optimal value of player* **max** *in game $G(x)$ is given by*

$$v_{\mathbf{max}}(x) = \sup_{\sigma} \inf_{\tau} v_{\sigma,\tau}(x) \ .$$

Similar, the optimal value of player **min** *in $G(x)$ is defined as*

$$v_{\mathbf{min}}(x) = \inf_{\tau} \sup_{\sigma} v_{\sigma,\tau}(x) \ .$$

Optimal strategy for a player is a strategy with which he can guarantee the optimal winning probability regardless of how the oponent plays.

Definition 8 (optimal strategies) *Strategy σ is optimal for player* **max** *in game $G(x)$ iff*

$$v_{\sigma,\tau}(x) \geqslant v_{\mathbf{max}}(x)$$

for any strategy τ of player **min**. *Similar, strategy τ is optiomal for player* **min** *in $G(x)$ iff*

$$v_{\sigma,\tau}(x) \leqslant v_{\mathbf{min}}(x)$$

for any strategy σ of player **max**.

The following fact is an easy consequence of the definition of optimal values.

Proposition 1 *For any game $v_{\mathbf{max}}(x) \leqslant v_{\mathbf{min}}(x)$*

PROOF: For any strategy τ' of player **min** we have $\inf_\tau v_{\sigma,\tau}(x) \leqslant v_{\sigma,\tau'}(x)$ and hence $\sup_\sigma \inf_\tau v_{\sigma,\tau}(x) \leqslant \sup_\sigma v_{\sigma,\tau'}(x)$. As τ' was chosen arbitrarily we get $\sup_\sigma \inf_\tau v_{\sigma,\tau}(x) \leqslant \inf_\tau \sup_\sigma v_{\sigma,\tau}(x)$, i.e. $v_{\mathbf{max}}(x) \leqslant v_{\mathbf{min}}(x)$. □

If the optimal values of both players are equal then we say that the game has a value.

Definition 9 (value of a game) *If $v_{\mathbf{min}}(x) = v_{\mathbf{max}}(x)$ then we say that the game has value $v_{\mathrm{opt}}(x)$ which is the common optimal value of both players.*

Observe that the notion of optimal value depends on the class of strategies which are considered. Restricting the class of strategies of a player, e.g., to only positional ones, can influence the best winning probabilities and whether the optimal value of a game exists or not. Here we consider arbitrary strategies with full memory because for them it is easy to prove existence of optimal values. It is also known that the optimal value for such games exists in positional strategies and is equal to the optimal value of general strategies. We shall prove this later for a class of terminating games (cf. Corollary 1).

If game has a value, then it is realized when both players play using their optimal strategies.

Proposition 2 *Suppose that game $G(x)$ has value. If σ is an optimal strategy for player* **max** *in the game and τ is optimal strategy for player* **min** *in the same game then*

$$v_{\sigma,\tau}(x) = v_{\mathrm{opt}}(x)$$

PROOF: From the fact that σ is optimal it follows that $v_{\sigma,\tau}(x) \geqslant v_{\mathbf{max}}(x) = v_{\text{opt}}(x)$ and from the fact that τ is optimal we get $v_{\sigma,\tau}(x) \leqslant v_{\mathbf{min}}(x) = v_{\text{opt}}(x)$. $\square$

Although in general the edge probabilities of a game board can be arbitrary, in practice they must have a finite representation. When considering algorithms which take game board as an input we will assume that all the edge probabilities are given by rational numbers of known precision.

Definition 10 (game board with rational data) *Game board G with N positions and out-degree bounded by K is a (N,K,D)-board if each edge probability $p(x,y)$ is a rational number of the form r/q with $r,q \in \mathbb{Z}$, $0 \leqslant r \leqslant q \leqslant D$.*

Note that a (N,K,D)-game board can be represented using space $N \cdot K \cdot \lg(D)$. This number will be taken as the size of the board. A special case are games played on $(N,2,2)$-boards, i.e., games in which each position has at most 2 equally probable successors. These are the simple stochastic games considered in [4, 5].

3 Successive Approximation

The successive approximation method on which we build our algorithms is based on the early result of Shapley [13] who shows that the optimal values of an infinite game can be approximated by the optimal values of truncated games in which only finitely many moves can be done.

We recall here Shapley's argument adapted to our setting. Let $G(x)$ be a stochastic reachability game. We consider a truncated game $G^n(x)$ which is played on game board G for at most n steps. The first player wins a play if he reaches the target within these n steps. Otherwise the second player is the winner.

The optimal value $v^n(x)$ of $G^n(x)$ is defined, as usual, to be the common value $\sup_\sigma \inf_\tau v^n_{\sigma,\tau}(x) = \inf_\tau \sup_\sigma v^n_{\sigma,\tau}(x)$. Since game $G^n(x)$ is finite, its optimal value exists and can be determined using the mini-max principle.

Proposition 3 *Let $v^n(x)$ be the optimal value of $G^n(x)$ and let*

$$v^{n+1}(x) = \begin{cases} 1 & \text{if } x = t, \\ \max_{x\to y} v^n(y) & \text{if } x \in V_{\max}, \\ \min_{x\to y} v^n(y) & \text{if } x \in V_{\min}, \\ \sum_{x\to y} p(x,y)\cdot v^n(y) & \text{if } x \in V_{\text{rnd}}. \end{cases} \tag{1}$$

Then $v^{n+1}(x)$ is the optimal value of $G^{n+1}(x)$.

Thus for any n we have a valuation $v^n : V \to [0,1]$ which assigns to each position x the optimal value $v^n(x)$ of the truncated game $G^n(x)$. As we shall show this sequence converges point-wise to the optimal values of the full game. We adapt Shapley's result to our setting of stochastic reachability games. Our proof is more general as it works also for nonterminating games.

Theorem 1 *Let G be a stochastic reachability game board. For any x the value of $G(x)$ exists and is equal to $\lim_{n\to\infty} v^n(x)$ where $v^n(x)$ is the value of the truncated game $G^n(x)$.*

PROOF: We first show that sequence v^n must converge point-wise to some valuation v^*. For any strategies σ and τ, the probability of reaching the target in $n+1$ steps is bigger than the probability of reaching it in n steps. Therefore $v^{n+1}_{\sigma,\tau}(x) \geqslant v^n_{\sigma,\tau}(x)$ for any σ and τ which implies that $v^{n+1}(x) \geqslant v^n(x)$. Hence, for each x the sequence $v^n(x)$ is bounded and nondecreasing and thus must converge to some value $v_*(x)$.

Next, we want to prove that

$$v_*(x) = \sup_\sigma \inf_\tau v_{\sigma,\tau}(x) = \inf_\tau \sup_\sigma v_{\sigma,\tau}(x) \ .$$

Note that, in the limit, the probabilities $v^n_{\sigma,\tau}(x)$ of reaching the target in at most n steps converge to the probability $v_{\sigma,\tau}(x)$ of reaching it in any number of steps.

Since $v^n(x) = \sup_\sigma \inf_\tau v^n_{\sigma,\tau}(x)$ for any n, we get

$$v_*(x) = \lim_{n\to\infty} \sup_\sigma \inf_\tau v^n_{\sigma,\tau}(x) = \sup_\sigma \inf_\tau \lim_{n\to\infty} v^n_{\sigma,\tau}(x) = \sup_\sigma \inf_\tau v_{\sigma,\tau}(x)$$

Similar, using the fact that $v^n(x) = \inf_\tau \sup_\sigma v^n_{\sigma,\tau}(x)$, we can prove $v_*(x) = \inf_\tau \sup_\sigma v_{\sigma,\tau}(x)$. $\square$

An important consequence of Theorem 1 and Proposition 3 is that the optimal values of a stochastic reachability game can be characterized as a solution of a system of equations which we call the *local optimality equations* (c.f. [4]).

Proposition 4 *Let $v_{\text{opt}}(x)$ be the optimal value of game $G(x)$ with target t. Valuation v_{opt} satisfies the following equations.*

$$\left\{\begin{array}{ll} v_{\text{opt}}(t) = 1 & \\ v_{\text{opt}}(x) = \max_{x\to y} v_{\text{opt}}(y) & x \in V_{\max}, \\ v_{\text{opt}}(x) = \min_{x\to y} v_{\text{opt}}(y) & x \in V_{\min}, \\ v_{\text{opt}}(x) = \sum_{x\to y} p(x,y) \cdot v_{\text{opt}}(y) & x \in V_{\text{rnd}}. \end{array}\right\} \tag{LOE}$$

PROOF: Above equations are obtained by taking limits in equations (1) and using the fact that $\lim_{n\to\infty} v^n = v_{\mathrm{opt}}$. $\square$

3.1 The basic algorithm

Theorem 1 and Proposition 3 are the basis of the value iteration method. Using Proposition 3 one can compute values v^n recursively, starting with v^0 – the optimal values for the game in which no moves can be made. Clearly, $v^0(t) = 1$ and $v^0(x) = 0$ for $x \neq t$. For $n > 0$, optimal values of $G^n(x)$ can be computed using recursive formulas (1). By Theorem 1, the computed values will converge to the optimal values of the game.

Example 1 Consider the game presented in Figure 1. In this game the optimal

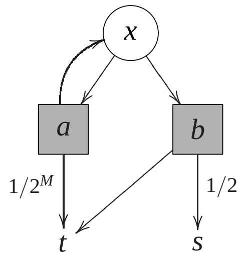

Figure 1: Example reachability game with one strategic position x belonging to player **max** and two random positions a and b. The probabilities of the edges $a \to x$ and $b \to t$ are $1 - 1/2^M$ and $1/2$, respectively.

strategy of player **max** is to play $x \to a$ after which the token must eventually reach target t with probability 1. Playing $x \to b$ will ensure reaching the target with probability $1/2$ only. Thus the optimal value of the game starting from x equals $v_{\mathrm{opt}}(x) = v_{\mathrm{opt}}(a) = v_{\mathrm{opt}}(t) = 1$ while $v_{\mathrm{opt}}(b) = 1/2$ and $v_{\mathrm{opt}}(s) = 0$.

Let us look at the first few truncated optimal values v^n computed using formulas (1) (we do not show values $v^n(t) = 1$ and $v^n(s) = 0$).

n	$v^n(x)$	$v^n(a)$	$v^n(b)$
0	0	0	0
1	0	$1/2^M$	$1/2$
2	$1/2$	$1/2^M$	$1/2$
3	$1/2$	$1/2 + 1/2^{M+1}$	$1/2$

The value $v^n(b)$ equals the optimal value $1/2$ already after one iteration. However, the values $v^n(a)$ and $v^n(x)$ converge very slowly to the optimal values. It can be shown that, e.g., $v_n(a) < 3/4$ for first $o(2^M)$ iterations.

Already for the third approximation we have $v^3(a) > v^3(b)$ which suggests that move $x \to a$ is a better choice for player **max** than $x \to b$. If we were able to verify that $x \to a$ is the optimal strategy, we could terminate the algorithm after 3 iterations and compute the optimal values using this strategy. ■

To turn the value iteration method into a working algorithm, one needs a suitable termination criteria which will tell when one can end computing successive approximations and determine the optimal values of the game.

Such criteria can be given for a class of *terminating* games in which infinite plays have zero probability. As we shall show, under this assumption one can give estimates for the number of steps needed to approximate optimal values with a given precision. Thus the iterations can be terminated after that many steps with the guarantee that computed approximations are close to optimal values. If the approximations are close enough, it is possible to reconstruct exact optimal values from them.

However, as suggested in Example 1, a better termination criteria is possible, based on *greedy* strategies which instruct each player to pick a move to a position with best value computed so far in the approximation process. It can happen that this way we can find optimal strategies early in the iteration process. We will elaborate on this shortly.

Definition 11 (greedy strategies) *Let v be a valuation of game positions. A v-greedy strategy for player* **max** *is a positional strategy σ which at $x \in V_{\max}$ chooses the successor y with the biggest value $v(y)$. A* **min** *player's positional strategy τ is v-greedy if it selects at $x \in V_{\min}$ the successor with the least value $v(x)$.*

Recall that $v_{\sigma,\tau}(x)$ is the probability of reaching the target when players use strategies σ and τ.

Definition 12 (stable strategies) *Positional strategies σ and τ are* stable *if they are $v_{\sigma,\tau}$-greedy.*

As we shall show in the next section, for terminating games stable strategies are optimal. It is also possible to determine in a polynomial time if a given pair of positional strategies is stable. This leads to the following algorithm for computing optimal strategies and values in stochastic reachability games.

Algorithm 1 (Successive Approximation)

1. *Set $v(t) := 1$, $v(x) := 0$ for $x \neq t$.*
2. *Let σ and τ be some v-greedy strategies.*

3. *Stop if σ and τ are stable, return $v_{\sigma,\tau}$.*
4. *Update v using recursive formulas (1), i.e., set $v := v'$ where*

$$v'(x) = \begin{cases} 1 & \text{if } x = t, \\ \max_{x\to y} v(y) & \text{if } x \in V_{\max}, \\ \min_{x\to y} v(y) & \text{if } x \in V_{\min}, \\ \sum_{x\to y} p(x,y)\cdot v(y) & \text{if } x \in V_{\mathrm{rnd}}. \end{cases}$$

5. *If σ or τ are not v-greedy then update them accordingly.*
6. *Repeat from 3.*

The algorithm computes successive approximations of the optimal values using recursive formulas (1). At the same time a pair of strategies σ and τ, greedy with respect to the current approximation, is maintained. Correctness of this algorithm relies on the fact that when strategies σ and τ are stable then they are also optimal and the corresponding winning probabilities $v_{\sigma,\tau}$ are the same as the optimal values of the game. In what follows we shall prove this fact for a class of terminating games. It is open whether the algorithm, as formulated here, also works correctly for non-terminating games.

3.2 Terminating Games

Games in which infinite plays have zero probability have special properties which will help us to prove correctness of the successive approximation algorithm presented above and to analyse its complexity. In particular, we can show that in such games stable strategies are optimal.

Definition 13 (terminating game) *Game $G(x)$ is terminating if for any strategies σ and τ and any starting position x the set of infinite plays of $G(x)$ conforming to σ and τ has probability 0. Game board G is terminating if $G(x)$ is terminating for any starting position x.*

Although the definition of a terminating game refers to arbitrary strategies and the probability distributions over the infinite set of plays of a game, there is a finite criteria for checking this property.

Proposition 5 *Game $G(x)$ is terminating iff for any positional strategies σ and τ there is a path conforming to σ and τ connecting x with a terminal position (a position which has no successors).*

Since there are only finitely many positional strategies on a given game board, a simple graph reachability algorithm can be used to check if a game is terminating or not.

We know that optimal values of a stochastic reachability game are a solution of the local optimality equations (LOE). In general these equations can have more than one solution but we shall prove that for terminating games the solution is unique.

In what follows we will often view a solution of the local optimality equations as a fixpoint of an operator corresponding to the right-hand sides of these equations. For a game board G with target position t, operator $F_G : (V \to [0,1]) \to (V \to [0,1])$ transforms a valuation v into a new valuation $F_G(v)$ given by

$$F_G(v)\,x = \begin{cases} 1 & x = t, \\ \max_{x\to y} v(y) & x \in V_{\max}, \\ \min_{x\to y} v(y) & x \in V_{\min}, \\ \sum_{x\to y} p(x,y)\cdot v(y) & x \in V_{\mathrm{rnd}}. \end{cases} \tag{2}$$

We call F_G an *update operator* for game board G and omit subscript G when it is clear from the context.

Now we can rephrase Proposition 4 and say that the optimal values of a game form a fixpoint of the update operator. Also note that equations (1) can be written as $v^{n+1} = F_G(v^n)$.

One can measure distance between two valuations of game positions using norm

$$\|v\| = \max_{x\in V} |v(x)|$$

for $v : V \to [0,1]$. It is well known that the set $V \to [0,1]$ equipped with $\|\cdot\|$ forms a Banach space (complete normed space). This will allow us to use Banach fixpoint theorem to prove that the fixpoint of the update operator acting on this space is unique.

The distance between two valuations v and w is taken to be $\|v - w\|$. Note that $\|v - w\| = 0$ iff $v = w$. To apply Banach fixpoint theorem we must show that the update operator is contracting, that is, it decreases the distance between two valuations to which it is applied. This is true for terminating game boards.

Lemma 1 *Let G be a terminating game board with N positions and the least edge probability m. For any valuations v and w we have*

$$\|F^N(v) - F^N(w)\| \leqslant (1 - m^N)\cdot \|v - w\|$$

where F is the update operator corresponding to G and $F^N(v)$ is an N-fold application of F to valuation v.

PROOF: Let us fix valuations v and w. We start the proof by constructing positional strategies σ and τ for players **max** and **min**, respectively. Then we will show that for any position x

$$|F^k(v)x - F^k(w)x| \leqslant (1 - s^{k-1}(x)) \cdot \|v - w\| \tag{i}$$

where $s^k(x)$ is the probability that a play starting at x and conforming to σ and τ terminates (i.e. reaches a terminal position) after at most k turns. This will prove our claim because $s^{N-1}(x) \geqslant m^{N-1}$ for any x. This is because the game board is terminating and hence there is a path conforming to σ and τ which connects x with a terminal position. Thus the terminal position can be reached in a play of $G(x)$ conforming to σ and τ after at most $N-1$ turns. The probability of this play equals the product of the probabilities of all its moves and each move has probability bigger than m.

Now we proceed to constructing strategies σ and τ. At position $x \in V_{\max}$ player **max** first checks whether $F(v)x \geqslant F(w)x$. If this is the case then he moves to the successor y of x with maximal value $v(y)$. Otherwise he selects the successor with maximal $w(y)$. Similar, at $x \in V_{\min}$, player **min** checks if $F(v)x \leqslant F(w)x$. Then he selects successor y with minimal value $v(y)$ if this is the case or with minimal value $w(y)$ otherwise.

Note that if x is a strategic position and its successor y is chosen according to the above strategies then

$$|F(v)x - F(w)x| \leqslant |v(y) - w(y)| \tag{ii}$$

For instance, if $x \in V_{\max}$ and $F(v)x \geqslant F(w)x$ then y is such that $v(y) = \max_{x \to y'} v(y')$ and hence

$$|F(v)x - F(w)x| = \max_{x \to y'} v(y') - \max_{x \to y'} w(y') \leqslant v(y) - w(y) = |v(y) - w(y)|$$

Similar arguments work in other cases.

We prove (i) by induction on k. Let T be the set of terminal positions of the game board G. Note that if $x \in T$ then $F^k(v)x = F^k(w)x$ for $k \geqslant 1$. This is because both values equal 1 in case x is the target and 0 otherwise. Hence (i) holds for $x \in T$.

Let $x \notin T$ be a strategic position. By (ii) and induction hypothesis,

$$|F^k(v)x - F^k(w)x| \leqslant |F^{k-1}(v)y - F^{k-1}(w)y| \leqslant (1 - s^{k-2}(y))\|v - w\|$$

where y is the successor of x chosen according to σ and τ. Since any play starting at x and conforming to σ and τ must go through y, $s^{k-1}(x) = s^{k-2}(y)$. Thus we have proven (i) for $k > 1$. If $k = 1$ then from (ii) and the fact that $s^0(x) = 0$,

$$|F(v)x - F(w)x| \leqslant |v(y) - w(y)| \leqslant (1 - s^0(x))\|v - w\|$$

Now consider the case of a random position $x \notin T$. Using induction hypothesis and the fact that $F^{k-1}(v)y = F^{k-1}(w)y$ for $y \in T$ and $k > 1$ we get

$$\begin{aligned}|F^k(v)x - F^k(w)x| &= |\sum_{y \notin T} p(x,y) \cdot (F^{k-1}(v)y - F^{k-1}(w)y)| \\ &\leqslant \sum_{y \notin T} p(x,y) \cdot (1 - s^{k-2}(y)) \cdot \|v - w\| \\ &= (1 - s^1(x) - \sum_{y \notin T} p(x,y) \cdot s^{k-2}(y)) \cdot \|v - w\|\end{aligned}$$

Note that $\sum_{y \notin T} p(x,y) \cdot s^{k-2}(y)$ is the probability of terminating a play starting from x in at most $k-1$ and at least 2 turns, i.e. it is $s^{k-1}(x) - s^1(x)$. Hence we obtain (i) for $k > 1$. For $k = 1$ we have

$$|F(v)x - F(w)x| = |\sum_y p(x,y) \cdot (v(y) - w(y))| \leqslant \|v - w\| = (1 - s^0(x)) \cdot \|v - w\|$$

as $s^0(x) = 0$ for $x \notin T$. $\square$

As an immediate consequence of the above lemma we obtain the following result which is a variant of similar results proven in [13] and [4]. In fact it is a straightforward application of the Banach fixpoint theorem to the contracting update operator F_G.

Theorem 2 *If G is a terminating game board then the corresponding local optimality equations have unique solution.*

PROOF: Suppose that v and w are two solutions of the local optimality equations. Then they are fixpoints of the update operator F and hence also of F^N. Therefore $\|v - w\| = \|F^N(v) - F^N(w)\| \leqslant \alpha \cdot \|v - w\|$ with $\alpha < 1$. This is possible only when $\|v - w\| = 0$, i.e. $v = w$. $\square$

3.3 Computing winning probabilities

The results derived so far allow us to give a polynomial algorithm for computing winning probabilities $v_{\sigma,\tau}$ corresponding to a given pair of stationary strategies σ and τ.

Proposition 6 *Let G be a game board with target t, σ and τ be positional strategies for player* **min** *and* **max**, *respectively. Valuation $v_{\sigma,\tau}$ is the unique solution of the following system of linear equations.*

$$\left\{\begin{array}{ll} v_{\sigma,\tau}(t) = 1 & \\ v_{\sigma,\tau}(x) = v_{\sigma,\tau}(\sigma(x)) & x \in V_{\max}, \\ v_{\sigma,\tau}(x) = v_{\sigma,\tau}(\tau(x)) & x \in V_{\min}, \\ v_{\sigma,\tau}(x) = \sum_{x \to y} p(x,y) \cdot v_{\sigma,\tau}(y) & x \in V_{\text{rnd}}. \end{array}\right\} \tag{3}$$

PROOF: Let $G^{\sigma,\tau}$ be the game board G restricted so that all moves which are not consistent with strategies σ and τ are removed. That is, in $G^{\sigma,\tau}$ every strategic position x has only one successor which is given by strategy σ or τ. Edges starting from random positions and their probabilities are the same in $G^{\sigma,\tau}$ and G.

On game board $G^{\sigma,\tau}$ players have no choice and σ and τ are the only possible strategies on that board. It follows that the optimal value of the game starting at any position x is the same as the winning probability $v_{\sigma,\tau}(x)$. It is also clear (by Proposition **??**) that if G is a terminating game board then so is $G^{\sigma,\tau}$. Hence we can apply Proposition 4 and Theorem 2 to game board $G^{\sigma,\tau}$ and conclude that $v_{\sigma,\tau}$ is the only solution of the corresponding local optimality equations, i.e., equations (3). $\square$

Since equations (3) are linear, they can be solved in a polynomial time, e.g., using the Gauss elimination method. Thus winning probabilities $v_{\sigma,\tau}$ for a given pair of stationary strategies σ and τ can be computed in polynomial time. Note that this also means that test for stable strategies which is required in step 3 of Algorithm 1 can be performed in a polynomial time: first one computes the winning probabilities $v_{\sigma,\tau}$ and then it can be checked in a linear time if strategies σ and τ are greedy with respect to these values.

We will denote by $F_{\sigma,\tau}$ the update operator corresponding to the restricted game board $G^{\sigma,\tau}$ and the right-hand sides of equations (3). It is a linear map given by

$$F_{\sigma,\tau}(v)x = \begin{cases} 1 & x = t, \\ v(\sigma(x)) & x \in V_{\max}, \\ v(\tau(x)) & x \in V_{\min}, \\ \sum_{x \to y} p(x,y) \cdot v(y) & x \in V_{\mathrm{rnd}}. \end{cases} \tag{4}$$

Above proposition states that the winning probabilities $v_{\sigma,\tau}$ are the unique fixpoint of this linear map.

The fact that the winning probabilities are the solution of a system of linear equations implies that they must be rational if the input data of a game is rational. The following result uses the same proof technique as in [4] where similar fact is proven for simple stochastic games.

Proposition 7 *Let G be a (N,K,D)-board. There exists integer $q \leqslant D^N$ such that for any position x the winning probability $v_{\sigma,\tau}(x)$ has the form r/q with $r \in \mathbb{Z}$, $0 \leqslant r \leqslant q$.*

PROOF: Equations (3) can be written in a matrix form as $v_{\sigma,\tau} = Q \cdot v_{\sigma,\tau} + b$ or $(I - Q) \cdot v_{\sigma,\tau} = b$. Multiplying both sides by constant D we can turn these equations into equations with integer coefficients. Let $A = D \cdot (I - Q)$ and $c = D \cdot b$. Then $v_{\sigma,\tau}$ is a unique solution of $A \cdot v_{\sigma,\tau} = c$.

From Cramer's rule we know that $v_{\sigma,\tau}(x) = \frac{\det A_x}{\det A}$ where A_x is matrix A with the column corresponding to x replaced by c. As matrices A and A_x contain only integer entries their determinants must be integer. Thus we can take $q = \det A$. It remains to show that $\det A \leqslant D^N$.

We use the fact that the determinant of a matrix equals the product of its eigenvalues. Any eigenvalue of A has the form $D \cdot \lambda$ where λ is an eigenvalue of $I - Q$. Moreover, any eigenvalue of $I - Q$ is of the form $1 - \lambda'$ where λ' is an eigenvalue of Q. Since Q is a sub-probabilistic matrix (contains only non-negative entries and each row sums up to a number smaller than 1) its eigenvalues are in the range $[0,1]$. Therefore $0 \leqslant \lambda \leqslant 1$ for any eigenvalue λ of $I - Q$ and thus all eigenvalues of A are bounded by D. It follows that $\det A \leqslant D^N$. $\square$

3.4 Correctness and complexity

The correctness of Algorithm 1 follows from the fact that for terminating games stable strategies are optimal. This is proven in [4] for the case of simple stochastic games. Here we present a slightly different proof based on the characterisation of the optimal values as a fixpoint of the update operator.

In our proof we use properties of pre and post fixpoints of the update operator considered as a map on the set of valuations equipped with the following partial order

$$v \sqsubseteq w \quad \Leftrightarrow \quad v(x) \leqslant w(x) \quad \text{for all } x \in V$$

where $v, w : V \to [0,1]$. This is a complete partial order on the set $V \to [0,1]$ and the update operator is a monotone map with respect to that order. Valuation $v : V \to [0,1]$ is a pre-fixpoint of the update operator F iff $F(v) \sqsubseteq v$. It is a post-fixpoint iff $v \sqsubseteq F(v)$. For a monotone operator it is known that any pre-fixpoint of F is bounded from below by the lest fixpoint of F and any post-fixpoint of F is bounded from above by the greatest fixpoint of F. In case F has only one fixpoint v_{opt}, it follows that $v_{\text{opt}} \sqsubseteq v$ for any pre-fixpoint v of F and $v \sqsubseteq v_{\text{opt}}$ for any post-fixpoint v.

Theorem 3 *Let G be a terminating game board. For any positional strategies σ and τ the following are equivalent:*

(a) strategies σ and τ are optimal,

(b) $v_{\sigma,\tau} = v_{\text{opt}}$.

(c) strategies σ and τ are stable,

(d) *strategies σ and τ are v_{opt}-greedy,*

PROOF: $(a) \Rightarrow (b)$: This implication was already proven as Proposition 2.

$(b) \Rightarrow (c)$: We use the fact that $v_{\sigma,\tau}$ and v_{opt} are fixpoints of operators $F_{\sigma,\tau}$ and F, respectively. Therefore (b) implies that

$$F_{\sigma,\tau}(v_{\sigma,\tau}) = v_{\sigma,\tau} = v_{\text{opt}} = F(v_{\text{opt}}) = F(v_{\sigma,\tau})$$

This means that strategies σ and τ are $v_{\sigma,\tau}$-greedy. For instance if $x \in V_{\max}$ then

$$v_{\sigma,\tau}(\sigma(x)) = F_{\sigma,\tau}(v_{\sigma,\tau})x = F(v_{\sigma,\tau})x = \max_{x \to y} v_{\sigma,\tau}(y)$$

$(c) \Rightarrow (d)$: First, we show that (c) implies (b). Indeed, if σ and τ are $v_{\sigma,\tau}$-greedy then $F(v_{\sigma,\tau}) = F_{\sigma,\tau}(v_{\sigma,\tau}) = v_{\sigma,\tau}$. Since v_{opt} is the only fixpoint of F it follows that $v_{\sigma,\tau} = v_{\text{opt}}$. Clearly, (c) together with (b) implies (d).

$(d) \Rightarrow (a)$: We prove that τ is optimal. Proof for σ is analogous.

Let σ' be arbitrary (not necessarily positional) strategy for player **max** and let $v_{\sigma',\tau}$ be the winning probabilities when players use strategies σ' and τ, respectively. Let σ'' be a positional, $v_{\sigma',\tau}$-greedy strategy. We will show that $v_{\sigma',\tau} \sqsubseteq v_{\sigma'',\tau}$ and $v_{\sigma'',\tau} \sqsubseteq v_{\text{opt}}$ which implies that τ is optimal because of arbitrary chosen σ'.

To show that $v_{\sigma'',\tau} \sqsubseteq v_{\text{opt}}$ we will prove that v_{opt} is a pre-fixpoint of $F_{\sigma'',\tau}$ (i.e., $F_{\sigma'',\tau}(v_{\text{opt}}) \sqsubseteq v_{\text{opt}}$). Since, for terminating game, $v_{\sigma'',\tau}$ is the only fixpoint of $F_{\sigma'',\tau}$ it follows that it bounds v_{opt} from below.

Let $x \in V_{\max}$. We have

$$F_{\sigma'',\tau}(v_{\text{opt}})x = v_{\text{opt}}(\sigma''(x)) \leqslant \max_{x \to y} v_{\text{opt}}(y) = F(v_{\text{opt}})x = v_{\text{opt}}(x)$$

Since τ is v_{opt}-greedy by our assumption, for $x \in V_{\min}$ we have

$$F_{\sigma'',\tau}(v_{\text{opt}})x = v_{\text{opt}}(\tau(x)) = \min_{x \to y} v_{\text{opt}}(y) = F(v_{\text{opt}})x = v_{\text{opt}}(x)$$

And for $x \in V_{\text{rnd}}$

$$F_{\sigma'',\tau}(v_{\text{opt}})x = \sum_{x \to y} p(x,y) \cdot v_{\text{opt}}(y) = F(v_{\text{opt}})x = v_{\text{opt}}(x)$$

which proves our claim that $F_{\sigma'',\tau}(v_{\text{opt}}) \sqsubseteq v_{\text{opt}}$.

We use similar technique to show that $v_{\sigma',\tau} \sqsubseteq v_{\sigma'',\tau}$. We will prove that $v_{\sigma',\tau}$ is a post-fixpoint of $F_{\sigma'',\tau}$ (i.e, $v_{\sigma',\tau} \sqsubseteq F_{\sigma'',\tau}(v_{\sigma',\tau})$) from which it follows that it is bounded from above by $v_{\sigma'',\tau}$, the only fixpoint of that operator.

Let $x \in V_{\min}$. The probability that a play conforming to σ' and τ and starting at x reaches target is the same as the probability that a game starting from $\tau(x)$ reaches the target. Thus

$$v_{\sigma',\tau}(x) = v_{\sigma',\tau}(\tau(x)) = F_{\sigma'',\tau}(v_{\sigma',\tau})x$$

If $x \in V_{\max}$ then the probability that a play starting at x and conforming to σ' and τ reaches target is the same as the probability that a game starting from the successor y which is chosen by strategy σ'. Since σ'' is $v_{\sigma',\tau}$-greedy we get

$$v_{\sigma',\tau}(x) = v_{\sigma',\tau}(y) \leqslant v_{\sigma',\tau}(\sigma''(x)) = F_{\sigma'',\tau}(v_{\sigma',\tau})x$$

Also in the case of $x \in V_{\mathrm{rnd}}$ we have

$$v_{\sigma',\tau}(x) = \sum_{x \to y} p(x,y) \cdot v_{\sigma',\tau}(y) = F_{\sigma'',\tau}(v_{\sigma',\tau})x$$

This shows that $v_{\sigma',\tau} \sqsubseteq F_{\sigma'',\tau}(v_{\sigma',\tau})$ and concludes our proof. □

We note the following consequence of the above theorem.

Corollary 1 *For any terminating stochastic reachability game there are optimal stationary strategies.*

PROOF: Since any game has optimal values v_{opt}, it is enough to take stationary v_{opt}-greedy strategies which are optimal by Theorem 3. □

The equivalences from the above theorem are valid only for terminating games as is illustrated by the following example.

Example 2 Consider the game from Figure 2. This is a non-terminating game since an infinite play is possible when players chose moves $x \to y$ and $y \to x$. Let σ be the strategy which at x selects the move to y and let τ be the strategy which moves from y to the target t. The corresponding winning probabilities are $v_{\sigma,\tau}(x) = v_{\sigma,\tau}(y) = 1$ and $v_{\sigma,\tau}(a) = 1/2$. Thus the strategies are $v_{\sigma,\tau}$-greedy, i.e., stable. However, strategy σ is not optimal, because **min** can play $y \to x$ preventing **max** from reaching the target. The optimal **max** move is to play $x \to a$ and ensure reaching the target with probability $1/2$. Thus the optimal values of the game are $v_{\mathrm{opt}}(x) = v_{\mathrm{opt}}(y) = v_{\mathrm{opt}}(a) = 1/2$. As we can see, in this non-terminating game stable strategies σ and τ are not optimal and $v_{\sigma,\tau} \neq v_{\mathrm{opt}}$. ■

The fact that for terminating game boards the corresponding update operator is contracting (Lemma 1) allow us to estimate the convergence rate of the sequence v^n and consequently give upper bounds on the number of iterations required by Algorithm 1.

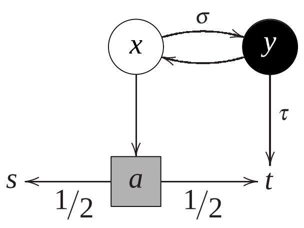

Figure 2: Stochastic reachability game with $x \in V_{\max}$ and $y \in V_{\min}$. From random position a there are equally probably moves to the target t and another sink s.

Lemma 2 *Let G be a terminating game board with N positions and the least edge probability m. Let $v^n(x)$ be the optimal value of the truncated game $G^n(x)$. If $n \geqslant N \cdot \lg(1/\varepsilon) \cdot (1/m)^N$ then $\|v^n - v_{\text{opt}}\| < \varepsilon$.*

PROOF: The optimal valuation v_{opt} is a fixpoint of F and hence also of F^N. Using Lemma 1 and the fact that $v^{n+1} = F(v^n)$ we can estimate, for $n > N$,

$$\|v^n - v_{\text{opt}}\| = \|F^N(v^{(n-N)}) - F^N(v_{\text{opt}})\| \leqslant \alpha \|v^{(n-N)} - v_{\text{opt}}\|$$

where $\alpha = (1 - m^{N-1})$. Thus for $n \geqslant kN$ we have

$$\|v^n - v_{\text{opt}}\| \leqslant \alpha^k \|v^{n-kN} - v_{\text{opt}}\| \leqslant \alpha^k$$

If $n \geqslant N \cdot \lg(1/\varepsilon) \cdot (1/m)^N$ then we can take $k = \frac{\lg(1/\varepsilon)}{m^N}$ in the above equation. Since $\lg(\alpha) = \lg(1 - m^{N-1}) < -m^{N-1} \leqslant -m^N$ we get $1/m^N > 1/-\lg(\alpha)$ and

$$k > \frac{\lg(1/\varepsilon)}{-\lg(\alpha)} = \frac{\lg(\varepsilon)}{\lg(\alpha)}$$

Hence $k \cdot \lg(\alpha) < \lg(\varepsilon)$, i.e., $\alpha^k < \varepsilon$. $\square$

Next we show that if the current approximation is close enough to the optimal values then the greedy strategies maintained by the algorithm must be optimal and stable, so that algorithm terminates.

Lemma 3 *Let G be a terminating (N, K, D)-board. If $v \sqsubseteq v_{\text{opt}}$ and $\|v - v_{\text{opt}}\| < D^{-N}$ then any v-greedy strategies are optimal and stable.*

PROOF: Assume that $v \sqsubseteq v_{\text{opt}}$, $\|v - v_{\text{opt}}\| < D^{-N}$ and that strategies σ and τ are v-greedy. We prove that they are also v_{opt}-greedy which, by Theorem 3, implies that they are optimal and stable.

First we note that if $v_{\text{opt}}(x) \neq v_{\text{opt}}(y)$ then they differ by at least $1/q \geqslant D^{-N}$. This is because, by Proposition 7, $v_{\text{opt}}(x) = p/q$ and $v_{\text{opt}}(y) = p'/q$ for some $q \leqslant D^N$ and $0 \leqslant p, p' \leqslant q$.

Suppose that σ is not v_{opt}-greedy at x. This means that $v_{\text{opt}}(y) > v_{\text{opt}}(\sigma(x))$ for some y. But then

$$v(y) > v_{\text{opt}}(y) - D^{-N} \geqslant v_{\text{opt}}(\sigma(x)) > v(\sigma(x))$$

which contradicts the fact that σ is v-greedy. Similar argument shows that τ is v_{opt}-greedy. $\square$

Above lemmas allow us to give an exponential upper bound on the number of iterations of the basic algorithm.

Theorem 4 *Let G be a terminating (N,K,D)-board. For such a game Algorithm 1 finds optimal strategies after at most $\lg(D) \cdot N^2 \cdot D^N$ iterations.*

PROOF: From Lemma 2 and the fact that the least edge probability m in G is bigger than $1/D$ we know that the distance between the current approximation v and v_{opt} is smaller than $\varepsilon = D^{-N}$ after no more than

$$N \cdot \lg(1/\varepsilon) \cdot (1/m)^N < N \cdot \lg(D^N) \cdot D^N = \lg(D) \cdot N^2 \cdot D^N$$

iterations. Thus the v-greedy strategies must by optimal, by Lemma 3. $\square$

4 Acceleration Schemas for Successive Approximation

Recall that a single step of the successive approximation algorithm consists of replacing the current approximation v by the new approximation $v' := F(v)$ where F is the update operator of the game. In order to terminate the computation, a pair of strategies σ and τ is maintained and updated so that they are greedy with respect to the current approximation. Algorithm terminates when it detects that strategies σ and τ are stable (and thus, in case of a terminating game, optimal).

Looking at example runs of the algorithm one can observe that in most iterations the greedy strategies σ and τ remain unchanged. Let us call replacing v by $v' = F(v)$ a *small-step* of the algorithm if the v-greedy strategies σ and τ are also v'-greedy so that they are not updated. Otherwise it is a *big-step* and the strategies must be updated. Only when a big-step happens there is a chance of terminating the computation while the role of small-steps is to lead from one big-step to another.

Our improvements of the basic approximation schema try to reduce the number of small-steps in the computation. This should substantially improve the convergence rate of the basic algorithm as one can observe that the small-steps are dominant in a typical run. Ideally, a big-step should be made in each iteration so that in each iteration the current strategies are updated and get closer to the optimal ones. This would make the algorithm similar to the strategy improvement method which also discovers a new strategy in each iteration. The difference is in the way the new strategies are discovered and in the computational cost of this operation.

In general our accelerated algorithms have the following structure.

Algorithm 2 (Accelerated Approximation)

1. *Set* $v(t) := 1$, $v(x) := 0$ *for* $x \neq t$.
2. σ *and* τ *are some* v*-greedy strategies.*
3. *Stop if* σ *and* τ *are stable.*
4. *Set* $v' := \mathsf{Improve}(v, \sigma, \tau)$ *and* $v := F(v')$.
5. *If* σ *or* τ *are not* v*-greedy then update them accordingly.*
6. *Repeat from 3.*

It is identical to that of the basic algorithm except that in line 4 the current approximation is improved before it is updated with F. Function $\mathsf{Improve}(v, \sigma, \tau)$ tries to predict how many small-steps will be done on v using strategies σ and τ and returns an approximate result of these updates. We propose three different implementations of $\mathsf{Improve}$ based on three different heuristics.

Before presenting details of our algorithms we give a general criteria which guarantees their correctness. It specifies requirements on function $\mathsf{Improve}$ which guarantee that the constructed sequence of approximations converges to the optimal values of the game.

Definition 14 (feasible valuation) *We say that valuation* v *is feasible at position* x *iff* $v(x) \leqslant F(v)x$. *Valuation* v *is feasible if it is feasible at every position in the game board (i.e.,* $v \sqsubseteq F(v)$ *and thus it is a post-fixpoint of* F*).*

Theorem 5 *If* $v' = \mathsf{Improve}(v, \sigma, \tau)$ *is feasible and satisfies* $v \sqsubseteq v'$ *for any feasible* v *and* v*-greedy* σ *and* τ *then Algorithm 2 computes optimal strategies for the game after at most* $\lg(D) \cdot N^2 \cdot D^N$ *many iterations when run on a* (N,K,D)*-board.*

PROOF: Let v_{opt} be the optimal valuation of the game and let v_i be its approximation after i iterations of Algorithm 2 (we set $v_0 = F(0)$).

Note that $v_{i+1} = F(v'_i)$ where v'_i is the valuation returned by the call to Improve in step 4 of the algorithm. Suppose that $v_i \sqsubseteq F(v_i)$. Then by our assumption about Improve, $v_i \sqsubseteq v'_i \sqsubseteq F(v'_i)$. Using the monotonicity of F we obtain

$$F(v_i) \sqsubseteq F(v'_i) = v_{i+1}$$

and

$$v_{i+1} = F(v'_i) \sqsubseteq F(F(v'_i)) = F(v_{i+1})$$

Using these inequalities and the fact that $v_0 \sqsubseteq F(v_0)$ we can prove by easy induction that

$$F(v_{i-1}) \sqsubseteq v_i \sqsubseteq F(v_i) \tag{*}$$

for all i. Second of these inequalities means that v_i is a post-fixpoint of the update operator F and hence it is bounded from above by v_{opt}, the only fixpoint of F. We conclude that $v_i \sqsubseteq v_{\text{opt}}$ for all i.

Let v^i be the i-th approximation computed by Algorithm 1 (i.e., $v^i(x)$ is the optimal value for the truncated game $G^i(x)$). Using the first inequality in (*) we can prove by induction that $v^i \leqslant v_i$ for all i. Thus $v^i \sqsubseteq v_i \sqsubseteq v_{\text{opt}}$ and we can estimate $\|v_{\text{opt}} - v_i\| \leqslant \|v_{\text{opt}} - v^i\|$. From Lemma 2 and Lemma 3 it follows that if $i > \lg(D) \cdot N^2 \cdot D^N$ then the v_i-greedy strategies must be optimal. $\square$

4.1 Acceleration I: Iterative Squaring

Our first acceleration technique is based on the observation that successive small-step updates of the current valuation v correspond to iterations of a linear map. For a linear map, cumulative effect of exponentially many iterations of it can be computed in a polynomial time using a method known as *iterative squaring*. We use this fact to decrease the number of consecutive small-step updates by an exponential factor.

Let us consider the current approximation v and the current v-greedy strategies σ and τ. We note that

$$F(v)x = \max_{x \to y} v(y) = v(\sigma(x))$$

for $x \in V_{\max}$ and

$$F(v)x = \min_{x \to y} v(y) = v(\tau(x))$$

for $x \in V_{\min}$. Thus applying F to v is the same as applying an update operator $F_{\sigma,\tau}$ corresponding to strategies σ and τ. Note that $F_{\sigma,\tau}$ can be written as $F_{\sigma,\tau}(v) = Q \cdot v + b$ where matrix Q has entries $q(x,y)$ given by

$$\begin{aligned} q(x,y) &= 1 && \text{if } x \in V_{\max} \text{ and } y = \sigma(x) \text{ or } x \in V_{\min} \text{ and } y = \tau(x), \\ q(x,y) &= p(x,y) && \text{if } x \in V_{\text{rnd}} \text{ and } (x,y) \in E, \\ q(x,y) &= 0 && \text{otherwise.} \end{aligned} \tag{5}$$

The base valuation b is given by $b(t) = 1$ and $b(x) = 0$ for all $x \neq t$.

As $F_{\sigma,\tau}(v) = Q \cdot v + b$, the result of 2 iterations of $F_{\sigma,\tau}$ is

$$F^2_{\sigma,\tau}(v) = F_{\sigma,\tau}(Q \cdot v + b) = Q^2 \cdot v + Q \cdot b + b = Q_1 \cdot v + b_1 \ .$$

where $Q_1 = Q^2$ and $b_1 = Q \cdot b + b$. Similar, the result of 4 iterations of $F_{\sigma,\tau}$ is

$$F^4_{\sigma,\tau}(v) = F^2_{\sigma,\tau}(Q_1 \cdot v + b_1) = Q_1^2 \cdot v + Q_1 \cdot b_1 + b_1 = Q_2 \cdot v + b_2 \ .$$

where $Q_2 = Q_1^2$ and $b_2 = Q_1 \cdot b_1 + b_1$. In general $F^{2^i}_{\sigma,\tau}(v) = Q_i \cdot v + b_i$ where $Q_0 = Q$, $b_0 = b$ and

$$\begin{aligned} Q_{i+1} &= Q_i^2 \\ b_{i+1} &= Q_i \cdot b_i + b_i \ . \end{aligned}$$

Thus at an expense of i matrix multiplications we can find the result of 2^i small-step updates of the current approximation v. This observation is used in our first approximation improvement function Improve1. The function uses the iterative squaring method to compute results of 2^i small-step updates of v for $i = 0, 1, 2, \ldots$. This is continued as long as strategies σ and τ remain greedy with respect to the current valuation.

```
Improve1$(v, \sigma, \tau) \equiv$
    $Q :=$ matrix given by equations (5);
    $b(t) := 1$;  $b(x) := 0$ for $x \neq t$;
    $v' := Q \cdot v + b$;
    **while** $\sigma$ and $\tau$ are $v'$-greedy **do**
      $v := v'$;
      $b := Q \cdot b + b$;  $Q := Q \cdot Q$;
      $v' := Q \cdot v + b$;
    **done**;
    **return** $v$.
```

Figure 3: Improvement using iterative squaring

We show that Improve1 satisfies premises of Theorem 5. Hence Algorithm 2 implemented using this function is correct.

Theorem 6 *Let $v' =$ Improve1(v, σ, τ) where v is feasible and σ and τ are v-greedy. Then v' is feasible and $v \sqsubseteq v'$.*

PROOF: From the construction of the function it follows that $v' = F^{2^k}_{\sigma,\tau}(v)$.

Since strategies σ and τ are v-greedy $F(v) = F_{\sigma,\tau}(v)$ and hence $v \sqsubseteq F_{\sigma,\tau}(v)$ by our assumption that $v \sqsubseteq F(v)$. By easy induction $v \sqsubseteq F^n_{\sigma,\tau}(v) \sqsubseteq F^{n+1}_{\sigma,\tau}(v)$ for all n.

Hence $v \sqsubseteq v'$ and $v' \sqsubseteq F_{\sigma,\tau}(v') = F(v')$. Last equality follows from the fact that strategies σ and τ are v'-greedy as is ensured by the function. $\square$

Thus we know that Algorithm 2 using function Improve1 finds the optimal strategies after at most exponentially many iterations. However, the cost of a single iteration is yet unknown as we do not know how many iterations can be done by the inner loop of the function. To investigate this issue we introduce the following notion.

Definition 15 ((σ,τ)-distance) *Let v be a valuation and σ, τ be some stationary strategies. The (σ,τ)-distance of v is the maximal possible number of small-step updates of v using strategies σ and τ. That is, it is the maximal number n such that σ and τ are $F^i(v)$-greedy for $i = 0, 1, \ldots, n$.*

The fact that each iteration of the while loop in Improve1 decreases the (σ,τ)-distance of the current valuation allows us to give bounds on the number of iterations.

Lemma 4 *Let n be the (σ,τ)-distance of v. A call to Improve1(v,σ,τ) returns result v' after at most $k \leqslant \lg(n)$ iterations of its inner while loop. Moreover,*

$$n' + 2^k \leqslant n < 2^{k+1}$$

where n' is the (σ,τ)-distance of v'.

PROOF: In the last, k-th iteration of the while loop, the current valuation v' is $F^{2^k}(v)$ and its (σ,τ)-distance is n'. Since strategies σ and τ are v'-greedy, the (σ,τ)-distance of v is at least $2^k + n'$. That is

$$n \geqslant n' + 2^k \geqslant 2^k \ .$$

It follows that the number of iterations is bounded by $\lg(n)$. The last iteration is terminated after discovering that strategies σ and τ are not $F^{2^k}(v')$-greedy. Hence $n < 2^{k+1}$. $\square$

On a terminating game board the small-step updates of a valuation converge to a limit valuation with a known convergence rate. This gives an exponential bound on a (σ,τ)-distance of a valuation.

Lemma 5 *Assume that G is a (N,K,D)-board. If strategies σ and τ are not optimal then the (σ,τ)-distance of a valuation $v \sqsubseteq v_{\sigma,\tau}$ is bounded by* $\lg(D) \cdot N^2 \cdot D^N$.

PROOF: Let n be the (σ,τ)-distance of v. As long as strategies σ and τ are v-greedy, applying F to v is the same as applying the linear operator $F_{\sigma,\tau}$ to it. Hence the sequence $F(v), F^2(v), \ldots, F^n(v)$ is the same as sequence

$$F_{\sigma,\tau}(v), F^2_{\sigma,\tau}(v), \ldots, F^n_{\sigma,\tau}(v)$$

which converges to $v_{\sigma,\tau}$.

Operator $F_{\sigma,\tau}$ is an update operator corresponding to the restricted game board $G^{\sigma,\tau}$ which is terminating. Hence we can use the arguments of Theorem 4 to show that if $n \geqslant \lg(D) \cdot N^2 \cdot D^N$ then $F^n_{\sigma,\tau}(v)$ is so close to $v_{\sigma,\tau}$ that any $F^n_{\sigma,\tau}(v)$-greedy strategies must be also $v_{\sigma,\tau}$-greedy. This would imply that σ and τ are $v_{\sigma,\tau}$-greedy and thus optimal by Theorem 3. Therefore n must be smaller than $\lg(D) \cdot N^2 \cdot D^N$. □

Using the above two lemmas we can prove the following bound on the number of iterations of the inner loop of Improve1.

Proposition 8 *Assume that G is a (N,K,D)-board. When function Improve1 is called in line 4 of Algorithm 2 then it returns after at most $O(N \cdot \lg(D))$ iterations of its inner while loop.*

PROOF: Note that Improve1 is called on (v,σ,τ) in Algorithm 2 only when σ and τ are not optimal. Moreover, $v \sqsubseteq F(v)$ and σ and τ are v-greedy.

It follows that $v \sqsubseteq v_{\sigma,\tau}$ and hence Lemma 5 can be applied to bound the (σ,τ)-distance of v by $\lg(D) \cdot N^2 \cdot D^N$. Since, by Lemma 4, the (σ,τ)-distance is halved in each iteration of the inner loop of Improve1 it follows that the number of iterations is bounded by $\lg\bigl(\lg(D) \cdot N^2 \cdot D^N\bigr) = O(N \cdot \lg(D))$. □

Similar observations lead to a linear bound on the number of successive small-step updates in the accelerated algorithm. This proves that using Improve1 substantially improves the basic algorithm in which an exponential number of small-steps is possible.

Theorem 7 *Accelerated algorithm with procedure Improve1 makes at most $O(N \cdot \lg(D))$ many consecutive small-steps when run an a (N,K,D)-board.*

PROOF: Let n be the (σ,τ)-distance of the current approximation v and let n' be the (σ,τ)-distance of the improved valuation v' returned as a result of a call to Improve1.

From Lemma 4 it follows that $n' \leqslant n/2$, i.e., the (σ,τ)-distance of the current approximation is at least halved in one iteration. If the iteration is a small-step then the strategies σ and τ remain greedy and the (σ,τ)-distance keeps decreasing with exponential rate. As the initial (σ,τ)-distance is bounded by $\lg(D)\cdot N^2\cdot D^N$ it follows that there can be at most $O(N\cdot\lg(D))$ consecutive small-step updates. $\square$

4.2 Acceleration II: Linear Extrapolation

Iterating operator $F_{\sigma,\tau}$ produces a sequence of valuations which converges to the values $v_{\sigma,\tau}$. Function Improve1 returns a valuation from this sequence. Thus the limit valuation $v_{\sigma,\tau}$ sets up a direction in which the current approximation is improved. As the limit valuation can be computed in a polynomial time, we propose to replace iterations of $F_{\sigma,\tau}$ by a linear extrapolation in the direction of $v_{\sigma,\tau}$. That is, we will improve the current approximation v by replacing it with a valuation

$$v_t = (1-t)\,v + t\,\tilde{v},$$

from the linear segment connecting v with some limit valuation $\tilde{v}$ (initially $\tilde{v} = v_{\sigma,\tau}$). Parameter $t \in [0,1]$ is chosen so that v_t is a valuation satisfying $v_t \sqsubseteq F(v_t)$. This will guarantee correctness of the accelerated algorithm using this improvement function. Of course we also want to maximize t so that the distance between v and the improved valuation v_t is as big as possible.

Function Improve2 uses recursive funciton Extrapolate accepting two parameters, the current feasible valuation v and a limit valuation $\tilde{v}$ in whose direction the current valuation will be extrapolated. Function Improve2 simply calls Extrapolate with $v_{\sigma,\tau}$ as the initial limit valuation (see Figure 4).

Givien initail, feasible valuation v, for each $x \in V$ we compute parameter t_x such that v_t is feasible at x for all $t \in [0,t_x]$ (that is, $v_t(x) \leqslant F(v_t)\,x$ for $t \in [0,t_x]$). If we take $t = \min_x t_x$ then v_t is feasible at all $x \in V$ and thus feasible.

Let $f_x(t) = v_t(x) - F(v_t)\,x$. Our assumption that v is feasible at x means that $f_x(0) \leqslant 0$ and we want to find t_x such that $f_x(t) \leqslant 0$ for $t \in [0,t_x]$.

In case x is a random position function $f_x(t)$ is linear in t. Thus inequality $f_x(t) \leqslant 0$ holds for all $t \in [0,t_x]$ where t_x is a solution of linear equation $f_x(t) = 0$.

If x is a strategic position, then $f_x(t) = v_t(x) - v_t(y)$ where y is one of the successors of x. We can consider only these successors y for which initial inequality $f_x(0) \leqslant 0$ holds, that is, such that $v(x) \leqslant v(y)$. For such y we have $v_t(x) \leqslant v_t(y)$ for $t \in [0,t^x_y]$ where t^x_y is a solution of linear equation $v_t(x) = v_t(y)$.

```
Improve2$(v,\sigma,\tau) \equiv$ Extrapolate$(v; v_{\sigma,\tau})$

Extrapolate$(v,\tilde{v}) \equiv$
    $t := 1$;
    **foreach** $x \in V$ **do**
        $\triangleright$ *Computing parameter* $t_x$ *and* $t = \min_x t_x$
        **if** $x \in V_{\mathrm{rnd}}$ **then** Let $t_x$ be a solution of $v_t(x) = F(v_t)x$.
        **else**
            **foreach** $(x,y) \in E$ s.t. $v(y) \geqslant v(x)$ **do**
                Find $t_y^x$ a solution to $v_t(x) = v_t(y)$.
            **if** $x \in V_{\min}$ **then** $t_x := \min_y t_y^x$ **else** $t_x := \max_y t_y^x$ **fi**
        **fi**
        **if** $t_x < t$ **then** $t := t_x$ **fi**
        $\triangleright$ *The limit valuation is updated if the position is blocked*
        **if** $t_x = 0$ **then** $\tilde{v}(x) := v(x)$ **fi**
    **done**;
    **if** $t > 0$ **then**
        **return** $v_t$
    **else**
        **return** Extrapolate$(v,\tilde{v})$
    **fi**
```

Figure 4: Improvement using linear extrapolation.

If $x \in V_{\max}$ we can take $t_x = \max_y t^x_y$. Then for $t \in [0, t_x]$

$$f_x(t) = v_t(x) - \max_{x \to y} v_t(y) \leqslant v_t(x) - v_t(y') \leqslant 0$$

where y' is the successor of x with maximal $t^x_{y'}$. If $x \in V_{\min}$ then we take $t_x = \min_y t^x_y$. In this case, for any $t \in [0, t_x]$ we have

$$f_x(t) = v_t(x) - \min_{x \to y} v_t(y) = v_t(x) - v_t(y') \leqslant 0$$

where y' is the successor of x with minimal value $v_t(y')$. The last inequality follows from the fact that $t \in [0, t^x_{y'}]$.

Above procedure finds parameters t_x and $t = \min_x t_x$ such that v_t is feasible. However, if one of t_x equals 0 then $t = 0$ and v_t is not different from the initial valuation v. Let us call position x *blocked* if $t_x = 0$. When a blocked position is detected by Extrapolate then it is unblocked by modifying the limit valuation. This way blocked positions are successively removed and eventually the function returns an improved valuation.

Proposition 9 *Assume that $v \sqsubsetneq F(v) \sqsubseteq \tilde{v}$. A call to Extrapolate$(v, \tilde{v})$ returns after making at most N recursive calls where N is the number of game positions. Moreover, if v' is the returned valuation then $v \sqsubsetneq v' \sqsubseteq F(v')$.*

PROOF: We first note that $v \sqsubseteq \tilde{v}$ implies $v \sqsubseteq v_t$ for any t as $v_t = (1-t)v + t\tilde{v}$. Next observe that if $\tilde{v}(x) = v(x)$ then

$$v_t(x) = v(x) \leqslant F(v)x \leqslant F(v_t)x$$

i.e., v_t is feasible at x for any t and thus $t_x = 1$. This means that position x such that $\tilde{v}(x) = v(x)$ can not be blocked. Therefore each recursive call reduces the number of blocked positions and hence there can be at most N recursive calls.

When there are no blocked positions then the returned valuation is v_t with $t = \min_x t_x$. As noted above $v \sqsubseteq v_t$. Also, v_t is feasible as $t \leqslant t_x$ for all x. Finally, $v_t \neq v$ since $t > 0$. □

We use Extrapolate to improve the current approximation in Algorithm 2. Function Improve2 simply calls Extrapolate with $v_{\sigma,\tau}$ as the initial limit valuation. We note that whenever Improve2 is invoked on (v, σ, τ) in the algorithm then $v \sqsubsetneq F(v)$ and $F(v) = F_{\sigma,\tau}(v) \sqsubseteq v_{\sigma,\tau}$ as strategies σ and τ are v-greedy. Therefore Improve2 satisfies premises of Theorem 5.

Theorem 8 *Let $v' =$ Improve2(v, σ, τ) where v is feasible and σ and τ are v-greedy. Then v' is feasible and $v \sqsubseteq v'$.*

PROOF: A direct consequence of Proposition 9. □

Using Theorem 5 we conclude that Algorithm 2 which uses function Improve2 correctly computes the optimal strategies for a game.

4.3 Acceleration III: Linear Programming

Our last acceleration technique uses linear programming to completely remove small-steps form the approximation schema. In this respect it is comparable to the strategy improvement method. A single iteration is relatively expensive to compute (although it is still polynomial) but we can guarantee that in each step a new, better pair of strategies is discovered.

Let us look again at the basic approximation algorithm. Let v be the current approximation with v-greedy strategies σ and τ. The result of n consecutive updates of v is the valuation $v_n = F^n(v)$. If the updates are small-steps then valuations v_n satisfy the following properties:

(a) $v \sqsubseteq v_n$,

(b) $v_n \sqsubseteq F(v_n)$ and

(c) strategies σ and τ are v_n-greedy.

We would be able to skip all the small-step updates of v if we find the maximal v_n which satisfies constraints (a)–(c).

We note that constraints (a) and (c) are linear. Moreover, when (c) is satisfied, constraint (b) is also linear. This is because if strategies σ and τ are v_n-greedy then $F(v_n) = F_{\sigma,\tau}(v_n)$ where $F_{\sigma,\tau}$ is the linear update operator corresponding to strategies σ and τ.

Therefore, in our last implementation of function Improve, we improve the current approximation by replacing it with a solution of a linear program corresponding to the constraints (a)–(c).
The valuation v' returned by a call to Improve3(v, σ, τ) does not necessarily coincide with a result of several small-step updates of v. However it satisfies the requirements of Theorem 5 and hence Improve3 can be correctly used in the accelerated algorithm.

Theorem 9 *Let $v' =$ Improve3(v, σ, τ) where v is feasible and σ and τ are v-greedy. Then v' is feasible and $v \sqsubseteq v'$.*

PROOF: First we note that under our assumptions valuation v satisfies constraints (a)–(c). Hence the solution v' of the linear program exists. We have $v \sqsubseteq v'$ since v' satisfies (a). From (b) and (c) it follows that $v' \sqsubseteq F_{\sigma,\tau}(v')$ and $F_{\sigma,\tau}(v') = F(v')$, thus $v' \sqsubseteq F(v')$. □

```
Improve3$(v,\sigma,\tau) \equiv$
    Return solution $v'$ of the linear program: maximize $\sum_x v'(x)$ subject to
        (a:  $v \sqsubseteq v'$)
             $v(x) \leqslant v'(x)$     for all $x$,
        (b:  $v' \sqsubseteq F_{\sigma,\tau}(v')$)
             $v'(x) \leqslant v'(\sigma(x))$     for $x \in V_{\max}$,
             $v'(x) \leqslant v'(\tau(x))$     for $x \in V_{\min}$,
             $v'(x) \leqslant \sum_{x \to y} p(x,y) \cdot v'(y)$     for $x \in V_{\mathrm{rnd}}$,
        (c:  $\sigma$ and $\tau$ are $v'$-greedy)
             $v'(y) \leqslant v'(\sigma(x))$     for $x \in V_{\max}$ and $(x,y) \in E$,
             $v'(\tau(x)) \leqslant v'(y)$     for $x \in V_{\min}$ and $(x,y) \in E$.
```

Figure 5: Improvement using linear programming.

We can prove that when using function Improve3 Algorithm 2 finds a new pair of strategies in each iteration, thus the small steps of the basic algorithm are completely removed. To show this we define a norm of a pair of strategies and prove that this norm is increased in each iteration of the algorithm.

Definition 16 (norm of strategies) *For positional strategies σ and τ let their norm $\|\sigma,\tau\|$ be the maximal value $\sum_x v(x)$ over the valuations satisfying constraints (b) and (c) of the linear program in function Improve3. That is, v should be such that $v \sqsubseteq F_{\sigma,\tau}(v)$ and strategies σ, τ are v-greedy.*

First we show that the valuation returned by Improve3(v,σ,τ) corresponds to the norm of σ and τ despite the fact that it satisfies an additional constraint (a).

Lemma 6 *Let $v' =$ Improve3(v,σ,τ) where v is feasible and σ and τ are v-greedy. Then $\sum_x v'(x) = \|\sigma,\tau\|$.*

PROOF: As v' satisfies constraints (b) and (c) we know that $\sum_x v'(x) \leqslant \|\sigma,\tau\|$.

Suppose that there is a valuation v'' satisfying constraints (b) and (c) such that $\sum_x v''(x) = \|\sigma,\tau\| > \sum_x v'(x)$. Then we can construct a new valuation v_* which assigns to x the bigger of the values $v'(x)$ and $v''(x)$. We argue that v_* satisfies (a)–(c) which is a contradiction as $\sum_x v_*(x) \geqslant \sum_x v''(x) > \sum_x v'(x)$.

It is obvious that v_* satisfies constraint (a). Let us take a position $x \in V_{\max}$. We know that $v'(x) \leqslant v'(\sigma(x))$ and $v''(x) \leqslant v''(\sigma(x))$ as both v' and v'' satisfy constraint (b). If $v_*(x) = v'(x)$ then $v_*(x) \leqslant v'(\sigma(x)) \leqslant v_*(\sigma(x))$ and similar argument works when $v_*(x) = v''(x)$. The same way we can show that v_* satisfies all the remaining inequalities using the fact that both v' and v'' satisfy them. $\square$

Now we are ready to prove that the norm of the current greedy strategies is increased in each iteration of the algorithm. This implies that the same pair of strategies is never repeated during the iterations.

Proposition 10 *Let (σ_i, τ_i) be the current greedy strategies in the i-th iteration of Algorithm 2 in which function* Improve3 *is used. Then $(\sigma_i, \tau_i) \neq (\sigma_j, \tau_j)$ for $j < i$.*

PROOF: We show that the sequence of strategies constructed by the algorithm is monotonic with respect to the norm $\|\cdot\|$.

Let $v' = \mathsf{Improve2}(v_i, \sigma_i, \tau_i)$ and $v'' = \mathsf{Improve2}(v_{i+1}, \sigma_{i+1}, \tau_{i+1})$. We have $v' \sqsubseteq F(v') = v_{i+1} \sqsubseteq v''$.

If $v' = v_{i+1}$ then v_{i+1} is a fixpoint of F and thus it equals the optimal valuation. In that case strategies σ_{i+1}, τ_{i+1} are optimal and therefore different from all the previous strategies.

If $v' \neq v_{i+1}$ then $\sum_x v'(x) < \sum_x v_{i+1}(x) \leqslant \sum_x v''(x)$. By Lemma 6 $\sum_x v'(x) = \|\sigma_i, \tau_i\|$ and $\sum_x v''(x) = \|\sigma_{i+1}, \tau_{i+1}\|$. Hence $\|\sigma_i, \tau_i\| < \|\sigma_{i+1}, \tau_{i+1}\|$. $\square$

5 Experimental Results

This section presents results of experiments in which we run and compare the algorithms described in this paper. We also compare our algorithms with ones based on strategy improvement method for solving stochastic games. We start this section with brief introduction of the strategy improvement method.

5.1 Strategy Improvement

Algorithms based on successive approximation work by updating values assigned to game positions. There is an alternative approach in which player strategies are successively updated until they become optimal. This method, known as strategy or policy improvement, was first applied by Derman [6] to Markov decisions processes which are one player stochastic games. Later, Hoffman and Karp [8] extended it to the setting of two player stochastic games. Condon analysed these algorithms in the context of simple stochastic games [5, 12]. They were also applied to stochastic reachability games with concurrent moves [3]. There are versions of strategy improvement method which can be applied to mean-payoff [2] and parity games [14].

The strategy improvement algorithm has the following general structure.

Algorithm 3 (Strategy Improvement)

1. *Start with arbitrary strategy τ for player* **min**.
2. *Compute optimal winning probabilities v_τ in one player game G^τ in which* **min** *choices are fixed by τ.*
3. *Stop if τ is v_τ-greedy.*
4. *Update τ so that it is v_τ-greedy.*
5. *Repeat from 2.*

In step 2, optimal winning probabilities in a one player stochastic reachability game must be computed. This task is equivalent to optimizing a Markov decision process. Below we will discuss two methods which can be used for that purpose.

Strategy update in step 4 consists of switching it at positions at which it is not greedy. Position $x \in V_{\min}$ is *switchable* if $v_\tau(y) < v_\tau(\tau(x))$ for some $x \to y$. Strategy τ is updated at every switchable position x so that it selects the successor of x with the least value of v_τ. If there are no switchable positions then strategy τ is stable. For terminating games this means that it is also optimal (cf. Theorem 3).

Termination of the algorithm is guaranteed by the fact that each improvement step produces a new, better strategy. It can be proven that if τ' is a result of switching τ at a switchable position x then $v_{\tau'} \sqsubseteq v_\tau$ and $v_{\tau'}(x) < v_\tau(x)$ (see e.g. [4]). This ensures that no strategy will be considered twice and hence the number of improvement steps is bounded by the total number of positional strategies for a player (which is exponential in the size of the game board).

In practice the number of iterations tends to be very small, usually growing linearly with the size of the game. It is a long standing open question whether there are polynomial bounds on the number of iterations of the strategy improvement method. Subexponential bounds on the expected number of improvements have been proven in [1]. Versions of the algorithm with strong subexponential bounds on the number of iterations have been presented in [9, 2].

5.1.1 Solving One Player Games

In step 2 of the strategy improvement algorithm the following problem must be solved: given a positional strategy τ for player **min** find the optimal value for player **max** in a game $G^\tau(x)$ in which choices of **min** are fixed by τ. Let us call it a τ-optimal value at x and denote it by $v_\tau(x)$.

The game G^τ is an instance of a Markov decision process controlled by player **max** whose goal is to maximize probability of reaching the target position. The optimal strategy for **max** can be found using the same strategy improvement method which originally was introduced by Derman to solve exactly this type of problems. In our setting of a two player game in which choices of one player are fixed by a given strategy the procedure takes the following form

```
Derman$(\tau) \equiv$
  1  Start with arbitrary strategy $\sigma$ for max.
  2. Compute winning probabilities $v_{\sigma,\tau}$.
  3. Stop if $\sigma$ is $v_{\sigma,\tau}$-greedy, return $v_{\sigma,\tau}$.
  4. Update $\sigma$ so that it is $v_{\sigma,\tau}$-greedy.
  5. Repeat from 2.
```

Recall that $v_{\sigma,\tau}$ can be computed in a polynomial time by solving linear equations (3). We will call a version of Algorithm 3 which uses procedure Derman in step 2 an *elementary strategy improvement*. The name emphasizes the fact that it is based on relatively simple computations. The drawback is that it is not know whether its subroutine Derman will always terminate within polynomially many iterations.

On the other hand, it is well known that the optimal values of a Markov decision processes can be computed in a polynomial time (see e.g. [7]). This is achieved via reduction to linear optimization problems. The optimal values v_τ corresponding to a given **min**'s strategy τ are solution of the following linear optimization problem:

Minimize $\Sigma_x v(x)$ subject to:

$$\begin{aligned} &v(t) = 1 \\ &v(x) = v(\tau(x)) && x \in V_{\min}, \\ &v(x) \geqslant v(y) && x \in V_{\max},\ (x,y) \in E, \\ &v(x) = \sum_{(x,y)\in E} p(x,y)\cdot v(y) && x \in V_{\text{rnd}}. \end{aligned} \tag{6}$$

Version of Algorithm 3 proposed by Hoffman and Karp determines winning probabilities v_τ in step 2 by solving above linear constraint problem 6. Since linear programs can be solved in a polynomial time ([10]), there is a polynomial bound on the complexity of a single iteration of the algorithm, although achieving this complexity requires rather involved computations. In contrast, the elementary variant of strategy improvement uses very simple computations in each iteration but there is no guarantee that a single improvement step can be completed in a polynomial time.

5.2 Experiments

The different algorithms solving stochastic reachability games trade the complexity of a single improvement step for the convergence rate measured by the number of iterations needed to solve a game. The most complex algorithms solve difficult linear optimization problems in hope of substantially reducing the number of iterations needed to find a solution. On the other hand, the basic value iteration and the elementary strategy improvement algorithms use rather simple computations in each step but they might require large number of iterations to finish their task. We expect our accelerated algorithms to lie somewhere in between these two extremes having a good convergence rate and a relatively low costs of a single iteration.

To verify our expectations we have run a small set of experiments in which random games were solved using different algorithms. The main objective was to see whether and how much the proposed acceleration methods reduce the number of iterations of the basic successive approximation schema. We have also compared our algorithms against the ones based on strategy improvement.

We want to emphasize that the quantity measured in our experiments is the number of iterations needed to solve a game. A final efficiency of an algorithm also depends on the cost of performing a single iteration and this varies considerably among different algorithms. Adequate measuring of this cost would require state-of-the-art, efficient implementations of all the required computations such as solving large systems of linear equation systems and linear optimization problems. We didn't address these issues as this is beyond the scope of our research. Instead, we have implemented all the algorithms using symbolic arithmetic on rational numbers to avoid rounding errors and we utilized existing linear programming libraries. This substantially simplified the implementation task but it also limited the size of games we can handle to several hundred positions.

The experiment consisted of generating over 470 random games of different sizes and solving each of them using the following algorithms:

A. basic successive approximation (Algorithm 1),

B. accelerated approximation using iterative squaring (Algorithm 2 using function Improve1 of Figure 3),

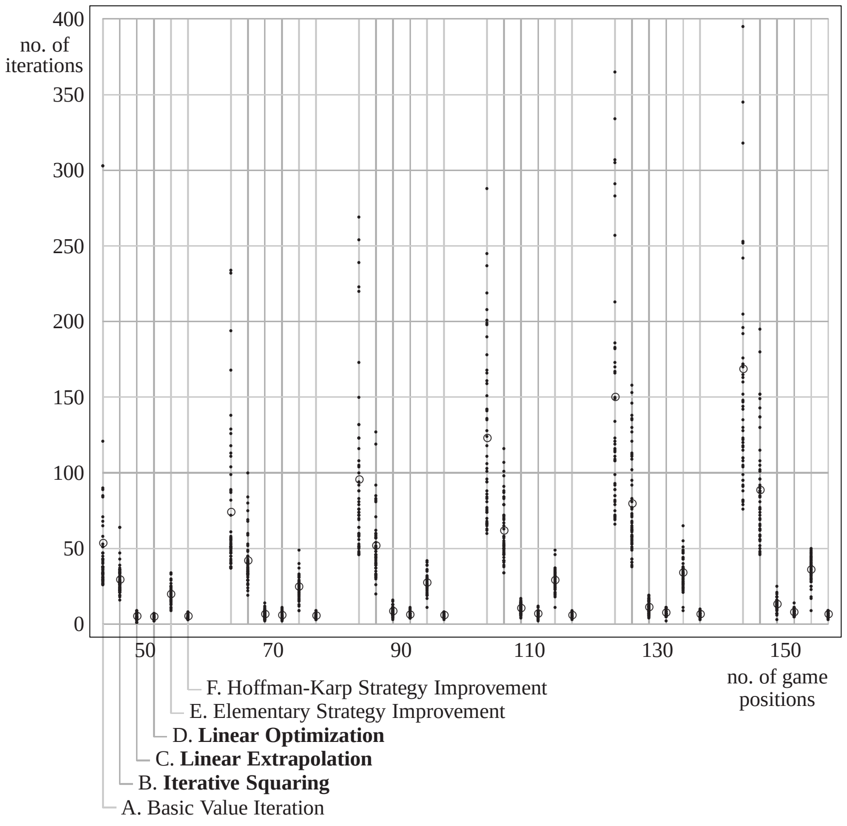

Figure 6: Number of iterations required to solve random simple stochastic games of differnet sizes by algorithms A–F. Each dot represents a single run of the corresponding algorithm on a game of the indicated size. Circles represent average number of iterations for games of the same size.

C. accelerated approximation using linear extrapolation (Algorithm 2 using function Improve2 of Figure 4),

D. accelerated approximation using linear optimization (Algorithm 2 using function Improve3 of Figure 5),

E. elementary strategy improvement (Algorithm 3 using function Derman),

F. Hoffman-Karp strategy improvement (Algorithm 3 using linear optimization).

The notion of a single iteration was different for different algorithms. In algorithms A,C and D it was a single update of the current valuation corresponding

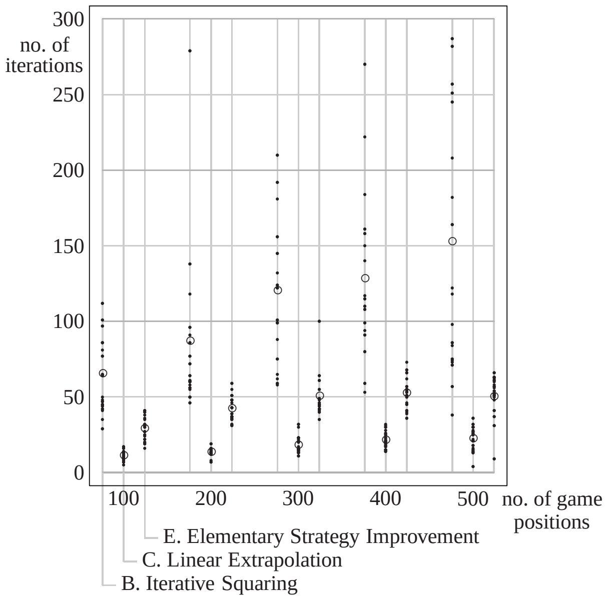

Figure 7: Results for algorithms B, C and E on games with 100 to 500 positions. Each dot represents a single run of the corresponding algorithm on a game of the indicated size. Circles represent average number of iterations for games of the same size.

to one iteration of the loop 3–6 in Algorithm 1 or Algorithm 2. For algorithm B we also counted each matrix multiplication as a separate iteration. Thus, if the inner loop of function Improve1 performed several iterations it was included in the global count. In algorithm E each update of one of the strategies counted as a single iteration. Finally, one iteration of algorithm F consists of determining the optimal values in the one-player game and switching the current strategy.

We used randomly generated simple stochastic games, that is, games in which each position has at most two successors and if it is a random position then both successors are equally probable. The number of positions ranged from 50 to 500. Due to heavy computations connected with solving linear optimization problems in algorithms D and F it was impractical to run them on games with more than 200 positions. For the simpler algorithms *B*, *C* and *D* we were able to run them on games with up to 500 positions.

Results of the experiments are presented in Figures 6 and 7. Figure 6 reports results from running all six algorithms on games with 50 to 150 positions. Approximately 50 games of each size were solved. Algorithms *B*,*C* and *E*

	A	B	C	D	E	F
A	–	6.01 1.83	53.00 12.97	65.29 17.72	16.26 3.94	91.40 19.21
B	0.94 0.60	–	28.00 7.02	31.75 9.54	8.22 2.14	32.50 10.37
C	0.30 0.10	0.40 0.17	–	4.50 1.46	1.22 0.33	4.17 1.59
D	0.24 0.08	0.35 0.13	3.33 0.83	–	0.67 0.24	2.75 1.14
E	1.11 0.35	2.05 0.57	13.00 3.71	10.00 4.75	–	8.33 4.85
F	0.30 0.08	0.40 0.12	4.00 0.80	2.67 1.00	0.33 0.21	–

A. Basic Value Approximation
B. Iterative Squaring
C. Linear Extrapolation
D. Linear Optimization
E. Elementary Strategy Improvement
F. Hoffman-Karp Strategy Improvement

Table 1: Maximal and average relative convergence rates of the algorithms.

were also run on games with 100 to 500 positions with around 30 games of each size solved. These results are shown in Figure 7.

Table 1 presents relative convergence rates of the algorithms. A relative convergence rate of algorithm X with respect to another algorithm Y is the ratio $\alpha = n_Y/n_X$ where n_X and n_Y are number of iterations needed by X and Y, respectively, to find a solution. Thus the ratio of 1 means that both algorithms used the same number of iterations. If $\alpha > 1$ then algorithm X used around α times less iterations than Y to solve the same game. In the table, the numbers in the column labelled by X and the row labelled by Y are the maximal and average relative convergence rate of X with respect to Y. Thus, for example, reading the first row of the table we can see that the Hoffman-Karp strategy improvement (algorithm F) required up to 90 times less iterations (20 times less on average) than the basic value iteration (algorithm A).

The basic idea of our acceleration methods was to reduce the number of small-steps, that is iterations which do not improve the current strategies. Table 2 shows the proportion of strategy updates in an average run of each of the algorithms based on the successive approximation scheme. Note that in case of algorithm D the measurements confirm that acceleration based on linear optimization eliminates small-steps entirely, as was proven in Proposition 10.

The results of the experiment show that accelerations based on linear extrapolation and linear optimization (algorithms C and D) substantially reduce the number of iterations as compared to the basic approximation scheme (algorithm A). In some cases they used up to 50–60 times less iterations and in

Basic Value Iteration	27%
Iterative Squaring	43%
Linear Extrapolation	97%
Linear Optimization	100%

Table 2: Number of strategy updates relative to the total number of iterations for the successive approximation algorithms.

average their convergence rate was around 15 times better than that of the basic algorithm A. However, the acceleration achieved using iterative squaring (algorithm B) was rather poor, probably not big enough to justify the high costs of a single iteration.

The experiments also show similarities between the acceleration based on linear optimization (algorithm D) and the Hoffman-Karp strategy improvement (algorithm F). Both algorithm use expensive computations in a single iteration and both have very similar convergence rates. It is interesting to see that comparable convergence rate was achieved by algorithm C using linear extrapolation. This acceleration method proves to be quite efficient, eliminating most of the small-step iterations while remaining relatively simple. A single iteration of algorithm C requires slightly more complicated computations than these used by the elementary strategy improvement (algorithm E) but the resulting convergence rate is approximately 3 times better. This makes algorithm C the most interesting alternative for the strategy improvement methods.

6 Conclusions and Future Work

We have demonstrated how the successive approximation scheme for solving stochastic games can be accelerated to substantially reduce the number of iterations needed to find optimal values and strategies. This leads to new interesting algorithms for solving stochastic reachability games. The algorithms are based on three different acceleration techniques which vary in the difficulty of the involved computations and in the results they achieve.

It is relatively cheap to execute the algorithm based on linear extrapolation. After computing the limit valuation, simple arithmetic operations are needed to linearly extend the current approximation in its direction. Despite its simplicity, in our experiments this algorithm proved to be very efficient, comparable to the most sophisticated ones based on linear constraint solving.

The algorithm using iterative squaring must perform matrix multiplications for matrices of the size proportional to the size of the game board. At this cost we can guarantee that it will update current greedy strategies after at most linearly many iterations. This is much better than for the basic, non-accelerated algorithm where exponentially many iterations might be necessary to alter the current greedy strategies.

The last algorithm needs to solve linear optimization problems in each iteration. This can be done in polynomial time, but is computationally rather involved. What we gain is that the current greedy strategies are updated in each iteration of the algorithm. This makes it similar to the strategy improvement method, where also a new strategy is found in each iteration and this also at the cost of solving linear constraint problems. What is different, is the way we arrive at the new strategy in each of the algorithms.

These theoretical considerations are confirmed by the experimental results we obtained by running our algorithms on randomly generated games. We could observe that the accelerated algorithms based on linear programming and linear extrapolation have similar efficiency as the strategy improvement one. We also noticed that our algorithms substantially increase frequency with which the greedy strategies are updated, with the one based on linear programming doing it in every iteration. In these experiments the algorithm using linear extrapolation looked especially attractive. As mentioned above it uses relatively simple computations in each iteration but it proved to be comparable to the most sophisticated algorithms based on linear constraint solving.

There is much work left to be done to better evaluate the usefulness of our new algorithms. First, a more efficient implementation would have to be constructed so that their performance can be verified on a more realistic sample of games. There is much space for improvement by using state-of-the-art techniques for efficient data representation, high precision arithmetic and fast solving of linear equation systems.

Currently we can prove correct behaviour of our algorithms only for a class of stopping games in which infinite plays have zero probability. It is known that any stochastic game can be approximated by a stopping one. However, such an approximation increases the size of the game by an exponential factor. It would be interesting to see if our algorithms can be made to directly work on arbitrary games, also these in which infinite plays have non-zero probability.

It is possible to give an interesting geometric interpretation of the way our accelerated algorithms work. Valuations of N game positions can be seen as points in the N-dimensional cube $[0,1]^N$. The algorithms search for the optimal values in a feasible region W within this cube. Our main observation is that region W can be divided into convex sub-regions $W_{\sigma,\tau}$ corresponding to positional strategies in the game. One way of looking at our algorithms is that they perform a search for the sub-region containing optimal values by moving from one sub-region to an adjacent one. They differ in the way this adjacent sub-region is found, but the cost of that operation is kept polynomial thanks to the fact that sub-regions are convex. Now it would be interesting to see if there are more efficient ways of performing this search in the set of sub-regions related to each other by their geometric location. The number of sub-regions is exponential, but the crucial question is if there is a polynomial path (computable in polynomial time) connecting an initial region with the one which contains optimal values.

References

[1] Henrik Björklund, Sven Sandberg, and Sergei G. Vorobyov. A discrete subexponential algorithm for parity games. In Helmut Alt and Michel Habib, editors, *STACS*, volume 2607 of *Lecture Notes in Computer Science*, pages 663–674. Springer, 2003.

[2] Henrik Björklund, Sven Sandberg, and Sergei G. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. In Jirí Fiala, Václav Koubek, and Jan Kratochvíl, editors, *MFCS*, volume 3153 of *Lecture Notes in Computer Science*, pages 673–685. Springer, 2004.

[3] Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Strategy improvement for concurrent reachability games. In *QEST*, pages 291–300. IEEE Computer Society, 2006.

[4] Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

[5] Anne Condon. On algorithms for simple stochastic games. In Jin-Yi Cai, editor, *Advances in Computational Complexity Theory*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 51–73. AMS, 1993.

[6] C. Derman. *Finite-State Markovian Decision Processes*. Springer, 1996.

[7] Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, New York, 1997.

[8] A. J. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management Science*, 12:359–370, 1966.

[9] Marcin Jurdzinski, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. Comput.*, 38(4):1519–1532, 2008.

[10] L.G. Khachiian. Polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.

[11] Walter Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117, 1995.

[12] Mary Melekopoglou and Anne Condon. On the complexity of the policy improvement algorithm. Technical Report 941, Univ. of Wisconsin – Madison, June 1990.

[13] L.S. Shapley. Stochastic games. *Proceedings of the National Academy of sceinces USA*, 39:1095–1100, 1953.

[14] Jens Vöge and Marcin Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *CAV'00: Computer-Aided Verification*, volume 1855 of *LNCS*, pages 202–215. Springer-Verlag, 2000.