Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science 119 (2005) 51–65

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

New Algorithms for Solving Simple Stochastic Games

Rafał Somla[1] ,[2]

Computing Science Department, Uppsala University, Box 337, SE-75105, Uppsala, Sweden

Abstract

We present new algorithms for determining optimal strategies for two-player games with probabilistic moves and reachability winning conditions. Such games, known as simple stochastic games, were extensively studied by A.Condon [2,3]. Many interesting problems, including parity games and hence also mu-calculus model checking, can be reduced to simple stochastic games. It is an open problem, whether simple stochastic games can be solved in polynomial time.
Our algorithms determine the optimal expected payoffs in the game. We use geometric interpretation of the search space as a subset of the hyper-cube $[0,1]^N$. The main idea is to divide this set into convex subregions in which linear optimization methods can be used. We show how one can proceed from one subregion to the other so that, eventually, a region containing the optinal payoffs will be found. The total number of subregions is exponential in the size of the game but, in practice, the algorithms need to visit only few of them to find a solution.
We believe that our new algorithms could provide new insights into the difficult problem of determining algorithmic complexity of simple stochastic games and other, equivallent problems.

Keywords: infinite graph games, parity games, simple stochastic games, finding optimal strategies, successive approximation.

1 Introduction

Many problems studied in computer science have an elegant presentation in a form of two-player graph games with various winning conditions. This includes verification of open components, controller synthesis and also theory of alternating automata. Hence a question of finding efficient algorithms for solving graph games, i.e., for deciding which player possess a winning strategy and

[1] Email: rsomla@it.uu.se

[2] This work is supported by Games for Processes project (VR grant 621–2002–455).

doi:10.1016/j.entcs.2004.07.008

possibly finding this strategy, becomes important. The problem was extensively studied for games with a wide range of winning conditions: from simple safety/reachability objectives to ω-regular ones expressed by either Büchi/co-Büchi or most general parity conditions [9,15,14,4,8,13,5]. A long standing open question in this area is whether it is possible to solve games with parity winning conditions in polynomial time. Through known reductions [12], a positive answer to this question would also mean that the mu-calculus model checking can be done in polynomial time.

In this paper we focus on *simple stochastic games* [2]. These are two-player, turn-based games with random moves. The objective in the game is to reach a final position (a sink) with the best possible associated payoff. Thus, rather than looking for a winning strategy, we want to find an *optimal strategy*, that is a strategy which guarantees the best expected payoff for a player.

We are interested in this kind of games because, on the one hand, other important graph games, like parity and mean-payoff games, can be easily reduced to simple stochastic games. On the other hand, simple stochastic games are instances of general stochastic games which have a rich and well developed theory. We believe that this link between an old area of operational research and the current studies can provide new insights into the problem of the complexity of graph games. Simple stochastic games are also interesting as a model for open, probabilistic components. Efficient algorithms for solving simple stochastic games can be used for verification of such components or even for synthesis of components meeting given specification.

Over the years, many algorithms has been proposed, which solve (simple) stochastic games. Many of them were later shown to be incorrect [3]. The correct ones, usually don't have any satisfactory complexity analysis. The two main methods used in these algorithms are the strategy improvement method and solving the local optimality equations.

Strategy improvement was developed by Hoffman and Karp for general stochastic games [7]. In this method an initial strategy for one of the players is improved in each iteration by switching it at positions at which choices are not locally optimal.

The other method is based on solving a system of constraints for the optimal expected payoffs in the game, which we call local optimality equations. Having optimal payoffs, one can easily reconstruct optimal strategies. For one-player games, the local optimality equations are linear, hence such games can be solved in polynomial time using linear programming techniques [6]. For two-player games the constraints are no longer linear and thus other methods are used, usually some form of iterative approximation.

We propose two algorithms which are based on the second method. For a game with N positions the vector of optimal expected payoffs is an element of a hyper-cube $[0, 1]^N$. Moreover, it is a maximal point of a set W of feasible vectors, described by the local optimality equations. Our main idea is to divide set W into subregions in which the equations become linear. This allows one to find in a polynomial time a maximal element in each subregion. Using this fact we show how to iterate through the subregions in such a way that eventually the subregion containing the optimal vector will be found. We prove that this must happen after at most an exponential number of iterations. In practice, we couldn't find any examples, including those known from the literature [3,10], which would require more than a polynomial number of iterations. To evaluate the efficiency of our algorithms we have implemented them and run on example simple stochastic games. In these test runs we show that our algorithms perform comparably to the strategy improvement methods.

The rest of this paper is organized as follows. Section 2 contains basic definitions and properties of simple stochastic games. The existence of the optimal value and memoryless determinacy of the game is stated here. We present first of our algorithms in Section 3. We prove its convergence to the optimal solution and also prove that the number of iterations of the algorithm is bounded by the number of different strategies in the game. Section 4 describes our second algorithm. It is a modification of the first algorithm which replaces costly solving of linear optimization problems by much simpler computations. We prove convergence of this simplified version of the algorithm and argue that the number of iterations is at most exponential in the size of the game. In Section 5 we describe the strategy improvement algorithms which we use as yardsticks to measure performance of our algorithms. Section 6 summarizes results of our experiments.

2 Simple Stochastic Games

Simple stochastic games are played by two players **max** and **min** on a *game board* G consisting of a finite directed graph of game positions. An edge in G indicates a possible move in the game. If a play reaches a sink s of G then it stops and player **max** wins from player **min** a payoff $p(s) \in [0, 1]$ associated with that sink.[3] Let S be the set of all sinks of G. The remaining positions V are divided into strategic and average (or random) ones. At a strategic position one of the players chooses the next move. Let $V_{\max}$ and $V_{\min}$ be

[3] In the standard definition of a simple stochastic game there are only two sinks—the 0-sink and the 1-sink. Player **max** wins a play if it ends in the 1-sink. Here we use a generalized version of the game (cf. [3, p. 53])

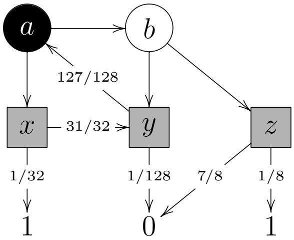

Fig. 1. Example game board. Black circles represent **min**-player nodes, white ones are **max**-player nodes. Gray squares represent average nodes. There are three sinks with payoffs 1, 0 and 1, respectively.

sets of positions where player **max** and **min**, respectively, makes a decision. At an average position $x \in V_{\text{avg}}$ the next move is chosen randomly with a given probability distribution $q(x, \cdot)$ over successors of x. Figure 1 presents an example game board with three average positions x, y and z, one **min** position a and one **max** position b. Three sinks on this board are labelled with their respective payoffs. Edges going out of average nodes are labelled with probabilities of their successors.

For each (nonterminal) position x game $G(x)$ starts at x and is played until a sink is reached. Hence a play of $G(x)$ is a maximal path $x, x_1, x_2, \ldots$ in the graph G. If it ends in a sink s then the outcome of the play is $p(s)$ and player **max** is interested in maximizing this outcome while **min** wants to minimize it.

A strategy for a player describes the choices which that player makes during a play of the game. In this paper we consider only deterministic, memoryless strategies which select the next move based on the current position, ignoring the history of a play.

Definition 2.1 [strategies] A strategy for a player $P \in \{\mathbf{max}, \mathbf{min}\}$ is a function $\sigma : V_P \to V \cup S$ such that $x \to \sigma(x)$ for all $x \in V_P$. A play $x_0, x_1, \ldots$ of the game $G(x_0)$ conforms to σ if $x_{i+1} = \sigma(x_i)$ for all $x_i \in V_P$.

In what follows, we will often consider valuations $v : V \to [0, 1]$ of game positions. When necessary, we extend such a valuation to game sinks with the payoff function p. The extension of v will be written as $\bar{v}$. Given a valuation we can speak about greedy strategies which make locally optimal choices with respect to that valuation.

Definition 2.2 [greedy strategies] Let $v : V \to \mathbb{R}$ be a node valuation. A **max** player strategy σ is v-greedy at $x \in V_{\max}$ if $\bar{v}\big(\sigma(x)\big) = \max_{x \to y} \bar{v}(y)$. A **min** player strategy τ is v-greedy at $x \in V_{\min}$ if $\bar{v}\big(\tau(x)\big) = \min_{x \to y} \bar{v}(y)$. Finally, a strategy for a player P is v-greedy if it is v-greedy at each $x \in V_P$.

Even when both players fix their strategies there are many possible plays due to the random choices made at average positions. Let σ and τ be strategies for players **max** and **min**, respectively. In a standard way edge probabilities induce a probability measure over plays conforming to σ and τ. Let $q_{\sigma,\tau}(x,s)$ be the probability that a play of $G(x)$ conforming to σ and τ ends in a sink s.

Definition 2.3 [expected payoffs] Let σ and τ be strategies for **max** and **min**, respectively. The expected payoff $v_{\sigma,\tau}(x) \in [0,1]$ in the game $G(x)$ when players use strategies σ and τ is defined by

$$v_{\sigma,\tau}(x) = \sum_{s \in S} q_{\sigma,\tau}(x,s) \cdot p(s).$$

Observe that implicit in this definition is the fact that an infinite play results in a payoff of 0.

Looking at the probability distribution over plays of $G(x)$ it is not hard to verify the following fact.

Proposition 2.4 *The vector $v_{\sigma,\tau}$ of expected payoffs is a fixed point of an operator $F_{\sigma,\tau} : (V \to [0,1]) \to (V \to [0,1])$ given by*

$$F_{\sigma,\tau}(v)\, x = \begin{cases} \bar{v}\big(\sigma(x)\big) & \text{if } x \in V_{\max}, \\ \bar{v}\big(\tau(x)\big) & \text{if } x \in V_{\min}, \\ \sum_{x \to y} q(x,y) \cdot \bar{v}(y) & \text{if } x \in V_{\text{avg}}. \end{cases}$$

Optimal strategies are defined in the usual way. It turns out that for simple stochastic games it is enough to consider memoryless strategies.

Definition 2.5 [optimal strategies/values] Strategies σ_*, τ_* are optimal at x if

$$v_{\sigma,\tau_*}(x) \le v_{\sigma_*,\tau_*}(x) \le v_{\sigma_*,\tau}(x)$$

for any σ and τ. The expected payoff $v_{\text{opt}}(x) = v_{\sigma_*,\tau_*}(x)$ is called an optimal value of the game $G(x)$ and is easily shown to be unique if it exists. Strategies σ and τ are optimal if they are optimal at every position in G.

The existence of the optimal value was proven by Shapley for general stochastic games [11] and later, by Condon, for the class of simple stochastic games [2]. The important assumption in these proofs is that a play of a game is finite with probability 1. [4]

[4] Using more advanced proof techniques, it is possible to obtain similar results also for non-stopping games. See e.g. a chapter on Positive Stochastic Games in [6].

Definition 2.6 [stopping game] G is a stopping game board if $\sum_{s \in S} q_{\sigma,\tau}(x, s) = 1$ for any x, σ and τ. That is, for any x, a play of $G(x)$ stops at a sink with probability 1 regardless of what strategies are used by the players.

The vector of optimal values of a stopping simple stochastic game is the only solution to the local optimality equations, as stated in the following theorem.

Theorem 2.7 (Shapley,Condon) *Let G be a stopping game board and let $F : (V \to [0,1]) \to (V \to [0,1])$ be given by*

$$F(v)\, x = \begin{cases} \max_{x \to y} \bar{v}(y) & \textit{if } x \in V_{\max}, \\ \min_{x \to y} \bar{v}(y) & \textit{if } x \in V_{\min}, \\ \sum_{x \to y} q(x,y) \cdot \bar{v}(y) & \textit{if } x \in V_{\text{avg}}. \end{cases}$$

The operator F has a unique fixed point v_* and $v_*(x)$ is the optimal value of $G(x)$ for all $x \in V$.

Proof (sketch). If G is a stopping game board with N non-terminal positions then the probability of reaching a sink in the first N steps of a play is at least m^N, where m is the least edge probability in G. This implies that an operator F^N is contracting, that is, $\|F^N(v) - F^N(w)\| \leq (1 - m^N)\, \|v - w\|$, where the norm $\|v\|$ of $v : V \to \mathbb{R}$ is defined by $\|v\| = \max_{x \in V} |v(x)|$. It follows that F^N, and hence also F, has a unique fixed point v_*.

Let σ_* and τ_* be v_*-greedy strategies. For any strategy σ for player **max** we have

$$F_{\sigma,\tau_*}\big(v_{\sigma_*,\tau_*}\big)\, x \leq F\big(v_{\sigma_*,\tau_*}\big)\, x = F_{\sigma_*,\tau_*}\big(v_{\sigma_*,\tau_*}\big)\, x = v_{\sigma_*,\tau_*}(x)$$

for all x. Observe that F_{σ,τ_*} is monotonic with respect to a partial order on $V \to [0,1]$ defined by $v \sqsubseteq v'$ iff $v(x) \leq v'(x)$ for all x. Hence, by Knaster-Tarski theorem, the unique fixed point of F_{σ,τ_*}, which is v_{σ,τ_*}, must be $\sqsubseteq$ than v_{σ_*,τ_*}. By similar argument we show that $v_{\sigma_*,\tau_*} \sqsubseteq v_{\sigma_*,\tau}$ for any **min** player strategy τ. This proves that strategies σ_* and τ_* are optimal. Since $v_{\sigma_*,\tau_*} = F_{\sigma_*,\tau_*}\big(v_{\sigma_*,\tau_*}\big) = F\big(v_{\sigma_*,\tau_*}\big)$ and F has a unique fixed point, it follows that $v_* = v_{\sigma_*,\tau_*}$. □

We note the following useful facts about optimal values and strategies.

Proposition 2.8 *Let G be a stopping game board and let $v_{\text{opt}}(x)$ be the optimal value of $G(x)$ for each x. The following are equivalent:*

(a) strategies σ and τ are optimal,

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

(c) *strategies σ and τ are $v_{\sigma,\tau}$-greedy.*

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

Proof. Implication $(a) \Rightarrow (b)$ follows by the uniqueness of the optimal value of a simple stochastic game. From (b) it follows that $F_{\sigma,\tau}(v_{\sigma,\tau}) = v_{\text{opt}} = F(v_{\sigma,\tau})$, i.e., σ and τ are $v_{\sigma,\tau}$-greedy. If (c) holds then $F(v_{\sigma,\tau}) = F_{\sigma,\tau}(v_{\sigma,\tau}) = v_{\sigma,\tau}$. Hence, by Theorem 2.7, $v_{\sigma,\tau} = v_{\text{opt}}$ and thus (d) also holds. Implication $(d) \Rightarrow (a)$ was proven as a part of the proof of Theorem 2.7. □

Implication $(d) \Rightarrow (a)$ of Proposition 2.8 shows how one can reconstruct optimal strategies knowing the optimal values of the game. The equivalence $(a) \Leftrightarrow (c)$ gives a polynomial procedure for deciding whether given strategies σ and τ are optimal: find the vector $v_{\sigma,\tau}$ by solving linear equations $v = F_{\sigma,\tau}(v)$ and then check if σ and τ are $v_{\sigma,\tau}$-greedy.

Corollary 2.9 *The problem of finding optimal strategies/values of a stopping simple stochastic game is in NP.*

From now on we restrict our attention to games played on a stopping game board.

3 Finding Optimal Values

From Theorem 2.7 we know that the vector of optimal values of a stopping game G is the unique fixed point of the operator

$$F(v)\,x = \begin{cases} \max_{x\to y} \bar{v}(y) & \text{if } x \in V_{\max}, \\ \min_{x\to y} \bar{v}(y) & \text{if } x \in V_{\min}, \\ \sum_{x\to y} q(x,y)\cdot\bar{v}(y) & \text{if } x \in V_{\text{avg}}. \end{cases}$$

Since F is $\sqsubseteq$-monotonic it follows, by Knaster-Tarski theorem, that its fixed point is a supremum of all the pre-fixed points of F. In other words, the vector of optimal values is a maximal point in a region W of the hyper-cube $[0,1]^V$ defined by

$$W = \big\{ v : V \to [0,1] \;\big|\; v \sqsubseteq F(v) \big\}.$$

In order to find this maximal point, we propose to divide W into subregions in which F is linear. This way the linear programming techniques can be used in each subregion to speed-up successive approximation of the optimal values.

For given strategies σ and τ, let

$$W_{\sigma,\tau} = \big\{ v \in W \;\big|\; \langle\sigma,\tau\rangle \text{ are } v\text{-greedy} \big\}.$$

Observe that F restricted to $W_{\sigma,\tau}$ is the same as $F_{\sigma,\tau}$ which is linear. Note also that different subregions $W_{\sigma,\tau}$ have disjoint interiors and that all the subregions sum up to W. The number of different sub-regions is the same as the number of different strategies and is exponential in the size of the game.

Our algorithm iterates through the subregions of W until it finds one corresponding to optimal strategies. In each iteration a new subregion is visited and this new subregion is determined using a maximal point of the current one. This maximal point is found by solving a linear program.

Algorithm 1 *(improved iteration I)*

1. *Start with* $v_1 = F(0)$.
2. *Find* v_i*-greedy strategies* $\langle\sigma_i, \tau_i\rangle$. *Stop if* $\langle\sigma_i, \tau_i\rangle$ *are optimal.*
3. *Find a valuation* v *which maximizes* $\sum_x v(x)$ *and satisfies the linear constraints:*
   (a) $v_i \sqsubseteq v$,
   (b) strategies $\langle\sigma_i, \tau_i\rangle$ *are* v*-greedy,*
   (c) $v \sqsubseteq F_{\sigma_i,\tau_i}(v)$.
4. *Take* $v_{i+1} = F(v)$ *and repeat.*

The convergence of the first algorithm to the optimal values easily follows from the following observations.

Lemma 3.1 *Suppose that* $v_n \in W$. *If* $v_n \sqsubseteq v \sqsubseteq F(v)$ *and* $v_{n+1} = F(v)$ *then* $F(v_n) \sqsubseteq v_{n+1}$ *and* $v_{n+1} \in W$.

Proof. This is a trivial consequence of F being monotonic: $F(v_n) \sqsubseteq v_{n+1}$ follows from $v_n \sqsubseteq v$ and $v_{n+1} \sqsubseteq F(v_{n+1})$ follows from $v \sqsubseteq v_{n+1} = F(v)$. □

Proposition 3.2 *If* $\{v_n\} \subseteq W$ *and* $F(v_n) \sqsubseteq v_{n+1}$ *then* $\lim_{n\to\infty} v_n = v_{\mathrm{opt}}$.

Proof. For each x the sequence $v_n(x)$ is bounded and monotonic, hence it converges to some value $v(x) \in [0, 1]$. Since $F(v_{n-1}) \sqsubseteq v_n \sqsubseteq F(v_n)$ it follows that $v = F(v)$ and therefore, by Theorem 2.7, $v = v_{\mathrm{opt}}$. □

Next, we show that the algorithm never visits the same subregion twice, from which it follows that it terminates after at most an exponential number of iterations.

Proposition 3.3 *Let* $v_0 \sqsubseteq v_1 \sqsubseteq v_2 \sqsubseteq \cdots \sqsubseteq v_n$ *be the sequence of valuations constructed by Algorithm 1 such that none of* v_i *is the optimal values vector. Let* W_i *be the subregion containing* v_i *and corresponding to the* v_i*-greedy strategies chosen in step 2 of the algorithm. Then* $W_i \neq W_j$ *for* $i \neq j$.

Proof. Suppose that $W_i = W_j = W_{\sigma,\tau}$ for some $i < j$. We have $v_{i+1} = F(v)$ and $v_{j+1} = F(v')$ where v and v' are maximal solutions to the same set of linear constraints. Moreover, $v \sqsubseteq v_{i+1} \sqsubseteq v_j \sqsubseteq v'$ and therefore $v = v'$. It follows that $v = v_{i+1} = F(v)$ so that v_{i+1} is the fixed point of F and hence, by Theorem 2.7, also the optimal values vector. This contradicts our assumption. □

Observe that the maximal points of the subregions, found in step 3 of the algorithm, form a $\sqsubseteq$-monotonic sequence. Therefore the sequence of subregions traversed by the algorithm is a chain in a partial order induced by $\sqsubseteq$-wise ordering of maximal points of these subregions. We expect that the maximal length of such a chain is much smaller than the total number of subregions, but so far we are unable to prove this formally.

4 Simplified Version

In each step of Algorithm 1 a linear optimization problem must be solved. This can be done in polynomial time but the solution can be costly to compute. In the next algorithm, we propose how to replace the linear optimization problem by much simpler computations at the expense of performing only sub-optimal improvements in each iteration.

Having the current valuation v_i and a pair of v_i-greedy strategies $\langle \sigma_i, \tau_i \rangle$ we compute $\tilde{v}_i = v_{\sigma_i,\tau_i}$ which we call the limit vector. This limit vector sets a direction in which the current valuation is increased.

If $\langle \sigma_i, \tau_i \rangle$ are $\tilde{v}_i$-greedy then, by Proposition 2.8, they are optimal and the algorithm can terminate. Otherwise, we look for the maximal point v_{t_*} in the segment $[v_i, \tilde{v}_i]$ such that $\langle \sigma_i, \tau_i \rangle$ are still v_{t_*}-greedy and we take $v_{i+1} = F(v_{t_*})$. Such a choice of v_{i+1} guarantees the convergence of the constructed sequence to v_{opt}.

Proposition 4.1 *Suppose that $v \in W_{\sigma,\tau}$ and that $v \sqsubseteq \tilde{v}$. Let $v_t = (1-t)\,v + t\,\tilde{v}$ for $t \in [0,1]$. The maximal t_* such that $v_{t_*} \in W_{\sigma,\tau}$ can be found in time $O(K)$ where K is the number of edges in the game graph.*

Proof. For each position x we find t_x such that strategy σ/τ is v_t-greedy at x for $t \in [0, t_x]$ in the following way. Let $x \in V_{\max}$ have successors $y_1, \ldots, y_k$ and $\sigma(x) = y_1$. Note that $v_0(y_1) \geq v_0(y_j)$ for all j, since σ is v-greedy at x. Inequality $v_t(y_1) \geq v_t(y_j)$ holds for all $t \in [0, t_j]$ where $t_j = \delta_j \cdot (\delta_j - \tilde{\delta}_j)^{-1}$, $\delta_j = v(y_1) - v(y_j)$ and $\tilde{\delta}_j = \tilde{v}(y_1) - \tilde{v}(y_j)$ (if $\delta_j = \tilde{\delta}_j$ we put $t_j = 1$). Strategy σ is v_t-greedy at x iff $v_t(y_1) \geq v_t(y_j)$ for all j. Hence $t_x = \min_j t_j$, and in the same way we can find t_x for $x \in V_{\min}$. Clearly $t_* = \min_x t_x$. □

This leads to the following algorithm.

Algorithm 2 *(improved iteration II)*

1. *Start with* $v_1 = F(0)$*.*

2. *Find* v_i*-greedy strategies* $\langle \sigma_i, \tau_i \rangle$ *and the limit vector* $\tilde{v}_i = v_{\sigma_i, \tau_i}$*. Stop if* $\langle \sigma_i, \tau_i \rangle$ *are* $\tilde{v}_i$*-greedy.*

3. *Let* $v_t = (1-t)\, v_i + t\, \tilde{v}_i$*. Find* t_**, the maximal* t *such that* $\langle \sigma_i, \tau_i \rangle$ *are* v_t*-greedy.*

4. *Take* $v_{i+1} = F(v_{t_*})$ *and repeat.*

The convergence of the constructed sequence of valuations to the optimal values follows from Proposition 3.2 and the following lemma

Lemma 4.2 *Let* $v \in W_{\sigma,\tau}$ *and* $\tilde{v} = v_{\sigma,\tau}$*. Let* $v_t = (1-t)\, v + t\, \tilde{v}$ *for* $t \in [0,1]$*. If strategies* $\langle \sigma, \tau \rangle$ *are* v_t*-greedy then* $v_t \in W_{\sigma,\tau}$*.*

Proof. We need to prove that $v_t \sqsubseteq F(v_t)$. The limit vector $\tilde{v} = v_{\sigma,\tau}$ is a fixed point of $F_{\sigma,\tau}$. From $v \in W_{\sigma,\tau}$ it follows that $F(v) = F_{\sigma,\tau}(v)$ and $v \sqsubseteq F(v)$. By linearity and monotonicity of $F_{\sigma,\tau}$ we get

$$v_t = (1-t)\, v + t\, \tilde{v} \sqsubseteq (1-t)\, F_{\sigma,\tau}(v) + t\, F_{\sigma,\tau}(\tilde{v}) = F_{\sigma,\tau}(v_t) = F(v_t)\ .$$

The last equality follows from the assumption that $\langle \sigma, \tau \rangle$ are v_t-greedy. □

Proposition 4.3 *Algorithm 2 finds an optimal pair of strategies after at most an exponential number of iterations.*

Proof. Let v_i be the sequence of valuations computed by Algorithm 2. Assuming $v_i \in W$ we see that $v_i \sqsubseteq v_{t_*}$ and, by Lemma 4.2, $v_{t_*} \sqsubseteq F(v_{t_*})$. Therefore, by monotonicity of F, $v_{i+1} = F(v_{t_*}) \sqsubseteq F(v_{i+1})$ and hence $v_{i+1} \in W$.

Clearly $v_1 = F(0) \in W$ thus $v_i \in W$ for all i. Also, from $v_i \sqsubseteq v_{t_*}$ it follows that $F(v_i) \sqsubseteq v_{i+1}$ for all i. By Proposition 3.2, $\lim_{i \to \infty} v_i = v_{\mathrm{opt}}$.

We have $F(v_i) \sqsubseteq v_{i+1} \sqsubseteq F(v_{i+1})$ and $v_1 = F(0)$, hence $F^i(0) \sqsubseteq v_i \sqsubseteq v_{\mathrm{opt}}$ for all i. Thus the convergence rate of v_i is no worse than the convergence rate of the sequence $F^i(0)$.

As noted in the proof of Theorem 2.7, operator F^N is α-contracting where N is the number of non-terminal positions in the game, $\alpha = (1-m)^N$ and m is the least edge probability in the game board. It follows that for $i = O\big((1/m)^N\big)$ the values $F^i(0)$, hence also v_i, are so close to the limit v_{opt} that any v_i-greedy strategies must be also v_{opt}-greedy and thus optimal. At that point the algorithm will terminate. □

Our experiments show that the number of iterations needed by Algorithm 2 to solve example games is slightly bigger but comparable with that of Algorithm 1. Hence, in practice, replacing the exact solution of the linear constraints problem by a heuristic choice of v_{t_*} seems to work well. Unfortunately, with this approach, the argument of Proposition 3.3 is no longer valid and, at least in principle, it is possible that Algorithm 2 traverses several times the same subregion during its search for the optimal values.

We note that a similar modification of the value iteration method is described in [1] in the context of Markov decision processes (i.e., single player stochastic games) as "generic rank-one corrections".[5] In this scheme, the current valuation is increased along a fixed vector d in each iteration, so that $v_{i+1} = F(v_i + \gamma d)$. However, the difficulty of this method lies in the correct choice of vector d which entails guessing the optimal values and correcting this guess during iterations.

In contrast, our algorithm uses the easily computable limit vector to determine the direction in which to improve the current valuation in each iteration. We also use a simple and general criteria (Proposition 3.2) which guarantees convergence of the modified sequence to the optimal values and works for single as well as two-player games.

5 Strategy Improvement Algorithms

None of the known methods for solving simple stochastic games has satisfactory complexity analysis. It seems though, that one of the simplest methods, the strategy improvement, works particularly well in practice. We decided to use strategy improvement algorithms as yardsticks to measure efficiency of our new algorithms.

The strategy improvement method is based on improving an initial, arbitrary strategy for one player, say **max**, by updating it at nodes at which it doesn't make optimal choices. Given a strategy σ, let $v_\sigma(x)$ be the best payoff player **max** can achieve in a game starting at x in which he uses σ. A position $x \in V_{\max}$ is *switchable* if there exists a successor y of x such that $v_\sigma(y) > v_\sigma(x)$. If this is the case, then the current strategy σ is updated to choose position y at x. After updating the current strategy at all switchable positions the whole process is repeated.

From this general schema we get different algorithms by using different methods for determining the values $v_\sigma(x)$. One method is to use the same strategy improvement technique to solve a one-player game resulting from fixing **max** choices according to the strategy σ. This leads to the following algorithm:

[5] I would like to thank the anonymous referee for pointing out this reference.

Algorithm 3 *(strategy improvement I)*

1. *Choose arbitrary strategies σ and τ for* **max** *and* **min**, *respectively.*
2. *Find an optimal* **min** *counter-strategy for σ by updating τ in the following process:*
   (a) *compute $v_{\sigma,\tau}$ by solving linear equations $v = F_{\sigma,\tau}(v)$,*
   (b) *for any* **min** *position x having a successor y such that $v_{\sigma,\tau}(y) < v_{\sigma,\tau}(x)$ set $\tau'(x) = y$,*
   (c) *set $\tau'(x) = \tau(x)$ for all remaining $x \in V_{\min}$,*
   (d) *if $\tau' \neq \tau$ then set $\tau \leftarrow \tau'$ and repeat from (b).*
3. *Update strategy σ based on the valuation $v_{\sigma,\tau}$. That is, for any $x \in V_{\max}$ if x has a successor y with $v_{\sigma,\tau}(y) > v_{\sigma,\tau}(x)$ then set $\sigma'(x) = y$. Otherwise set $\sigma'(x) = \sigma(x)$.*
4. *If $\sigma' = \sigma$ then strategies σ and τ are optimal. Otherwise set $\sigma \leftarrow \sigma'$ and repeat from 2.*

The advantage of this version of the algorithm is its simplicity. Besides solving the system of linear equations, one needs only to compare values of game positions and update strategies accordingly. On the other hand, nothing is known about the worst case complexity of this method. In principle, each improvement of the **max** strategy σ can cost exponentially many iterations of the inner loop in step 2. But in practice , despite its simplicity, the algorithm performs surprisingly well.

The other version of the algorithm uses linear programming to find payoffs $v_\sigma(x)$ in polynomial time. Determining the optimal values of a single-player stochastic game, also known as a Markov decision process, by solving a system of linear constraints is a well established technique which can be directly applied here.

Algorithm 4 *(strategy improvement II)*

1. *Choose arbitrary strategy σ for* **max**.
2. *Find expected payoffs v_σ for player* **max** *using strategy σ by solving the following optimization problem: maximize $\sum_x v_\sigma(x)$ under constraints*

$$\begin{aligned}
v_\sigma(x) &\le \bar{v}_\sigma(y) && \text{for } x \in V_{\min} \text{ and } x \to y,\\
v_\sigma(x) &= \bar{v}_\sigma\big(\sigma(x)\big) && \text{for } x \in V_{\max},\\
v_\sigma(x) &= \sum_{x \to y} q(x,y)\,\bar{v}_\sigma(y) && \text{for } x \in V_{\text{avg}},\\
v_\sigma(x) &\le 1 && \text{for all } x.
\end{aligned}$$

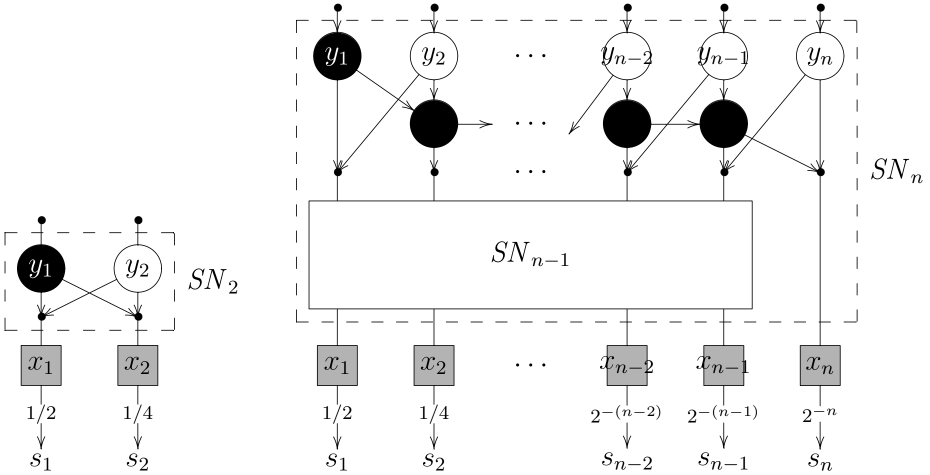

Fig. 2. Sorting games SG_n. For readibility edges connecting each average position x_i to the corresponding "output position" y_i were omited.

3. Update strategy σ to σ' based on the valuation v_σ found in 2.

4. If $\sigma' = \sigma$ then $v_\sigma = v_{\text{opt}}$. Otherwise set $\sigma \leftarrow \sigma'$ and repeat from 2.

As with our algorithms, each iteration of Algorithm 4 takes a polynomial time. But we note again, that the cost of solving linear constraint problems can be high in practice.

6 Experiments

We tested our algorithms on a family of simple stochastic games which we call sorting games. These games seem to be non-trivial for solving by various methods.

A sorting game board SG_n has n average positions $x_1, \ldots, x_n$ and n sinks $s_1, \ldots, s_n$. The strategic positions of SG_n are arranged into a sorting network SN_n which copies values of the average nodes to the "output" positions $y_1, \ldots, y_n$, sorting them into increasing order. Figure 2 shows how the sorting network SN_n is constructed from SN_{n-1}. As can be easily seen, network SN_n consists of $n(n-1)$ nodes and hence board SG_n has n^2 non-terminal positions and n sinks. Each average position x_i in SG_n is connected to the sink s_i with probability $1/2^i$ and to the corresponding position y_i with probability $1-1/2^i$.

For games SG_3, SG_4 and SG_5 we searched for sink payoffs which make the algorithms perform a large number of iterations. We abstracted from the details of the implementation of the algorithms and we compared only the number of iterations each algorithm needs to solve a game. This means that we treated solving a linear constraint problem as an atomic operation. For Algorithm 3 we counted the total number of strategy updates, taking into account also the costs of the inner loop of step 2.

Results of our experiments are presented in Table 1.

Table 1
Number of iterations taken by each of the algorithms on example sorting games.

ex. no.	no. of game positions	Alg. 1	Alg. 2	Alg. 3	Alg. 4
1	16	2	3	6	2
2	16	3	6	7	2
3	16	4	7	7	3
4	16	2	4	8	3
5	25	3	6	7	2
6	25	2	5	8	2
7	25	4	8	9	3
8	25	6	9	10	3
9	36	4	14	10	3
10	36	8	6	11	4
11	36	5	9	16	4
12	36	7	6	18	5

7 Summary

We present two algorithms for determining optimal values and strategies in a simple stochastic game. The algorithms search for the solution of the local optimality equations in the set W of feasible valuations. Search space W is divided into subregions $W_{\sigma,\tau}$ corresponding to different pairs of strategies. Our first algorithm uses linear optimization techniques to advance to a new subregion in each iteration. The second algorithm replaces exact solutions of the optimization problems by easily computable heuristics.

We prove correctness of both algorithms. A single iteration of each of the algorithms can be done in a polynomial time. Hence the complexity of the algorithms depends mainly on the number of iterations required to find the solution. We provide exponential bounds for this number and give some indications why it could be much smaller in practice.

We also tested our algorithms on example games and compared them with the well known method of strategy improvement. We demonstrate that in practice their performance is similar.

Our main contribution is a new technique for finding the optimal values of a simple stochastic game which is comparable with the strategy improvement method. The technique is based on a geometric interpretation of node valuations as points in the hyper-cube $[0, 1]^N$ and identification of player strategies with the subregions of this cube. In this setting we show a different method for improving the current pair of strategies by advancing from one subregion to the other in a monotonic way.

References

[1] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume II. Athena Scientific, 1995,2001.

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

[3] 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.

[4] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. In *Proc. 39th IEEE Symp. on Foundations of Computer Science*, pages 564–575, 1998.

[5] Luca de Alfaro and Rupak Majumdar. Quantitative solution of omega-regular games. In *ACM Symposium on Theory of Computing*, pages 675–683, 2001.

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

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

[8] Marcin Jurdziński. Small progress measures for solving parity games. In *Procedings of STACS*, volume 1770 of *LNCS*, pages 290–301, 2000.

[9] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

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

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

[12] Colin Stirling. Bisimulation, modal logic and model checking games. *L. J. of the IGPL*, 7(1), 1999.

[13] 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.

[14] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.

[15] Uri Zwick and Mike S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1–2):343–359, 1996.