Simple Stochastic Games with Few Random Vertices Are Easy to Solve*

Hugo Gimbert[1] and Florian Horn[2]

[1] LaBRI, Université Bordeaux 1, France
hugo.gimbert@labri.fr
[2] LIAFA, Université Paris 7, France
florian.horn@liafa.jussieu.fr

Abstract. We present a new algorithm for solving Simple Stochastic Games (SSGs). This algorithm is based on an exhaustive search of a special kind of positional optimal strategies, the *f-strategies*. The running time is $\mathcal{O}(\ |V_{\mathrm{R}}|! \cdot (|V||E| + |p|)\ )$, where $|V|, |V_{\mathrm{R}}|, |E|$ and $|p|$ are respectively the number of vertices, random vertices and edges, and the maximum bit-length of a transition probability.

Our algorithm improves existing algorithms for solving SSGs in three aspects. First, our algorithm performs well on SSGs with few random vertices, second it does not rely on linear or quadratic programming, third it applies to all SSGs, not only stopping SSGs.

1 Introduction

Simple Stochastic Games (SSGs for short) are played by two players Max and Min in a sequence of steps. Players move a pebble along edges of a directed graph (V, E). There are three type of vertices: V_{Max} is the set of vertices of player Max, V_{Min} the set of vertices of player Min and V_{R} the set of random vertices. When the pebble is on a vertex of V_{Max} or V_{Min}, the corresponding player chooses an outgoing edge and moves the pebble along it. When the pebble is on a random vertex, the successor is chosen randomly according to some fixed probability distribution: from vertex $v \in V_{\mathrm{R}}$ the pebble moves towards vertex $w \in V$ with some probability $p(w|v)$ and the probability that the game stops is 0, *i.e.* $\sum_{w \in V} p(w|v) = 1$. An SSG is depicted on Figure 1, with vertices of V_{Max} represented as $\bigcirc$, vertices of V_{Min} represented as $\square$, and vertices of V_{R} represented as $\Diamond$.

Player Max and Min have opposite goals, indeed player Max wants the pebble to reach a special vertex $t \in V$ called the *target vertex*, if this happens *the play is won by player* Max. In the opposite case, the play proceeds forever without reaching t and is *won by player* Min. For technical reasons, we assume that t is a vertex of player Max and is absorbing. *Strategies* are used by players to choose

* This research was partially supported by french project ANR "DOTS".

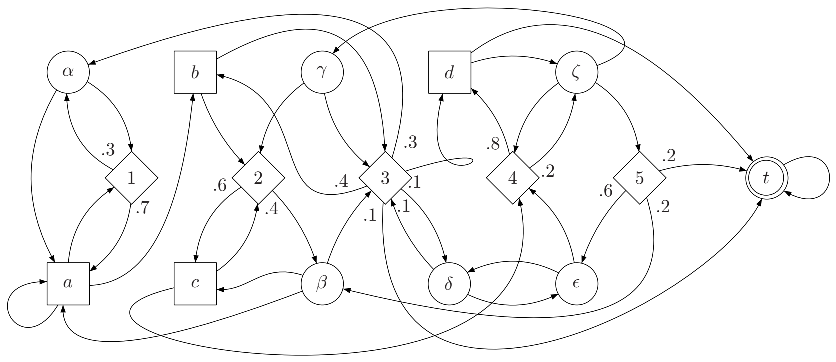

Fig. 1. A Simple Stochastic Game

their moves, a strategy tells where to move the pebble depending on the sequence of previous vertices, *i.e.* the finite path followed by the pebble from the beginning of the play. The *value* of a vertex v is the maximal probability with which player Max can enforce the play to reach the target vertex. When player Max, respectively player Min, uses an *optimal strategy* he ensures reaching the target with a probability greater, respectively smaller, than the value of the initial vertex. Notions of value and optimal strategies are formally defined in Section 2.

We are interesting in *solving* SSGs, that is computing values and optimal strategies.

Applications. SSGs are a natural model of reactive systems, and algorithms for solving SSGs may be used for synthesizing controllers of such systems. For example an hardware component may be modelled as an SSG whose vertices are the global states of the component, the target is some error state to avoid, random states are used to model failure probabilities and stochastic behaviours of the environment, choices of player Min corresponds to actions available to the software driver and player Max corresponds to non-deterministic behaviour of the environment. Then an optimal strategy for player Min in the SSG will correspond to a software driver minimizing the probability to enter the error state, whatever be the behaviour of the environment.

Existing algorithms for solving SSGs. The complexity of solving SSGs was first considered by Condon [Con92], who proved that deciding whether the value of an SSG is greater than $\frac{1}{2}$ is in NP $\cap$ co-NP. The algorithm provided in [Con92] consists in first transforming the input SSG in a *stopping SSG* where the probability to reach a sink vertex is 1. The transformation keeps unchanged the fact that the initial vertex has value strictly greater than $\frac{1}{2}$ but induces a quadratic blowup of the size of the SSG. The algorithm then non-deterministically guesses the values of vertices, which are rational numbers of linear size, and checks that these values are the unique solutions of some *local optimality equations*.

Three other kinds of algorithms for solving SSGs are presented in [Con93]. These algorithms require transformation of the initial SSG into an equivalent *stopping* SSG and are based on local optimality equations. First algorithm computes values of vertices using a quadratic program with linear constraints. Second algorithm computes iteratively from below the values of the SSGs, and the third is a strategy improvement algorithm *à la* Hoffman-Karp. These two last algorithms require solving an exponential number of linear programs, as it is the case for the algorithm recently proposed in [Som05].

Finally, these four algorithms suffer three main drawbacks.

First, these algorithms rely on solving either an exponential number of linear programs or a quadratic program, which may have prohibitive algorithmic cost and makes the implementation tedious.

Second, these algorithms only apply to the special class of *stopping* SSGs. Although it is possible to transform any SSG into a stopping SSG with arbitrarily small change of values, computing exact values this way requires to modify drastically the original SSG, introducing either $|V|^2$ new random vertices or new transition probabilities of bit-length quadratic in the original bit-length. This also makes the implementation tedious.

Third, the running time of these algorithms may *a priori* be exponential whatever be the number of random vertices of the input SSG, including the case of SSGs with no random vertices at all, also known as reachability games on graphs. However it is well-known that reachability games on graphs are solvable in quadratic time.

Notice that randomized algorithms do not perform much better since the best randomized algorithms [Lud95, Hal07] known so far run in sub-exponential expected time $e^{O(\sqrt{n})}$.

Our results. In this paper we present an algorithm that computes values and optimal strategies of an SSG in time $\mathcal{O}(\ |V_{\mathrm{R}}|! \cdot (|V||E| + |p|)\ )$, where $|V_{\mathrm{R}}|$ is the number of random vertices, $|V|$ is the number of vertices and $|p|$ is the maximal bit-length of transition probabilities.

The key point of our algorithm is the fact that optimal strategies may be looked for in a strict subset of positional strategies, called the class of **f**-strategies. The **f**-strategies are in correspondence with permutations of random vertices. Our algorithm does an exhaustive search of optimal **f**-strategies among the $|V_{\mathrm{R}}|!$ available ones and check their optimality. Optimality is easy to check, it consists in computing a reachability probability in a Markov Chain with V_{R} states, which amounts to solving a linear system with at most $|V_{\mathrm{R}}|$ equations.

Comparison with existing work. We improve existing results by three aspects: our algorithm performs better on SSGs with few random vertices, it is arguably much simpler, and we provide new insight about the structure of optimal strategies.

Our algorithm performs much better on SSGs with few random vertices than previously known algorithms. Indeed, its complexity is $\mathcal{O}(\ |V_{\mathrm{R}}|! \cdot (|V||E| + |p|)\ )$, hence when there are no random vertices at all, our algorithm matches the usual quadratic complexity for solving reachability games on graphs. When the number of random vertices is fixed, our algorithm performs in polynomial time, and on the class of SSGs such that $|V_{\mathrm{R}}| \leq \sqrt{|V_{\mathrm{Max}}| + |V_{\mathrm{Min}}|}$ our algorithm is sub-exponential.

Our algorithm is arguably simpler than previously known algorithms. Indeed, it does not require use of linear or quadratic programming. Although linear programs can be solved in polynomial time [Kac79, Ren88], this requires high-precision arithmetic. By contrast, our algorithm is very elementary: it enumerates permutations of the random vertices and for each permutation, it solves a linear system of equations.

Our algorithm is also simpler because it applies directly to any kind of SSGs, whereas previously known algorithms require the transformation of the input SSG into a stopping SSG of quadratic size.

Plan. The paper is organised as follows. In the first section, we introduce formally SSGs, values and optimal strategies. In the second section, we present the notion of **f**-strategies. In the third section, we focus on two properties of **f**-strategies: self-consistency and progressiveness. We prove that **f**-strategies that are both self-consistent and progressive are also optimal, and we prove the existence of such strategies. In the fourth section we describe our algorithm for solving SSGs.

Omitted proofs can be found in the full version of the paper [GH07].

2 Simple Stochastic Games

In this section we give formal definitions of an SSG, values and optimal strategies.

An SSG is a tuple $(V, V_{\text{Max}}, V_{\text{Min}}, V_{\text{R}}, E, t, p)$, where (V, E) is a graph, $(V_{\text{Max}}, V_{\text{Min}}, V_{\text{R}})$ is partition of V, $t \in V$ is the target vertex and for every $v \in V_{\text{R}}$ and $w \in V$, $p(w|v)$ is the transition probability from v to w, with the property $\sum_{w \in V} p(w|v) = 1$.

A *play* is an infinite sequence $v_0 v_1 \cdots \in V^\omega$ of vertices such that if $v_n \in (V_{\text{Max}} \cup V_{\text{Min}})$ then $(v_n, v_{n+1}) \in E$ and if $v_n \in V_{\text{R}}$ then $p(v_{n+1}|v_n) > 0$. A play is *won by* Max if it visits the target vertex; otherwise the play is won by Min. A *finite play* is a finite prefix of a play.

A *strategy* for player Max is a mapping $\sigma : V^* V_{\text{Max}} \to V$ such that for each finite play $h = v_0 \dots v_n$ such that $v_n \in V_{\text{Max}}$, we have $(v_n, \sigma(h)) \in E$. A play $v_0 v_1 \cdots$ is *consistent with* σ if for every n, if $v_n \in V_{\text{Max}}$ then v_{n+1} is $\sigma(v_0 \cdots v_n)$. A strategy for player Min is defined similarly, and is generally denoted τ.

Once the initial vertex v and two strategies σ, τ for player Max and Min are fixed, we can measure the probability that a given set of plays occurs. This probability measure is denoted $\mathbb{P}_v^{\sigma,\tau}$. For every $n \in \mathbb{N}$, we denote by V_n the random variable defined by $V_n(v_0 v_1 \cdots) = v_n$, the set of plays is equipped with the σ-algebra generated by random variables $(V_n)_{n \in \mathbb{N}}$. Then there exists a probability measure $\mathbb{P}_v^{\sigma,\tau}$ with the following properties:

$$\mathbb{P}_v^{\sigma,\tau}(V_0 = v) = 1 \tag{1}$$
$$\mathbb{P}_v^{\sigma,\tau}(V_{n+1} = \sigma(V_0 \cdots V_n) \mid V_n \in V_{\text{Max}}) = 1, \tag{2}$$
$$\mathbb{P}_v^{\sigma,\tau}(V_{n+1} = \tau(V_0 \cdots V_n) \mid V_n \in V_{\text{Min}}) = 1, \tag{3}$$
$$\mathbb{P}_v^{\sigma,\tau}(V_{n+1} \mid V_n \in V_{\text{R}}) = p(V_{n+1}|V_n). \tag{4}$$

Expectation of a real-valued, measurable and bounded function ϕ under $\mathbb{P}_v^{\sigma,\tau}$ is denoted $\mathbb{E}_v^{\sigma,\tau}[\phi]$. We will often use implicitly the following formula, which gives the expectation of ϕ once a finite prefix $h = v_0 v_1 \cdots v_n$ of the play is fixed:

$$\mathbb{E}_v^{\sigma,\tau} [\ \phi \mid V_0 \cdots V_n = h] = \mathbb{E}_{v_n}^{\sigma[h],\tau[h]} [\ \phi[h]\ ]\,, \tag{5}$$

where $\sigma[h](w_0 w_1 w_2 \cdots) = \sigma(v_0 \cdots v_n w_1 w_2 \cdots)$ and $\tau[h]$ and $\phi[h]$ are defined similarly.

Values and positional optimal strategies. The goal of player Max is to reach the target vertex t with the highest probability possible, whereas player Min has the opposite goal. Given a starting vertex v and a strategy σ for player Max, whatever strategy τ is chosen by Min, the target vertex t will be reached with probability at least:

$$\inf_\tau \mathbb{P}_v^{\sigma,\tau} (\text{Reach}(t))\,,$$

where Reach(t) is the event $\{\exists n \in \mathbb{N}, V_n = t\}$. Thus, starting from v, player Max can ensure to win the game with probability arbitrarily close to:

$$\text{val}_*(v) = \sup_\sigma \inf_\tau \mathbb{P}_v^{\sigma,\tau} (\text{Reach}(t))\,,$$

and symmetrically, player Min can ensure that player Max cannot win with a probability much higher than:

$$\text{val}^*(v) = \inf_\tau \sup_\sigma \mathbb{P}_v^{\sigma,\tau} (\text{Reach}(t))\,.$$

Clearly $\text{val}_*(v) \leq \text{val}^*(v)$. In fact these values are equal, and this common value is called the value of vertex v and denoted val(v). A much stronger result is known about SSGs: the infimum and supremum used in the definition of val(v) are attained for some strategies called *optimal strategies*. Moreover, there exists optimal strategies of a simple kind, called *positional strategies*. A strategy σ is said to be positional if it depends only on the current vertex, *i.e.* for every finite play $v_0 \cdots v_n$ $\sigma(v_0 \cdots v_n) = \sigma(v_n)$. The following results are well-known [Sha53, Con92].

Theorem 1. *In any SSG, for every vertex $v \in V$, the values* $\text{val}_*(v)$ *and* $\text{val}^*(v)$ *are equal. This common value is called the value of vertex v and denoted* val(v)*. There exists strategies $\sigma^\#$ and $\tau^\#$ that are* optimal *i.e. for every vertex v and every strategies σ, τ:*

$$\mathbb{P}_v^{\sigma,\tau^\#} (\text{Reach}(t)) \leq \text{val}(v) \leq \mathbb{P}_v^{\sigma^\#,\tau} (\text{Reach}(t))\,.$$

Moreover there exists strategies that are both optimal and positional.

3 Playing with f-Strategies

Existence of positional optimal strategies is a key property of SSGs, used for designing all known algorithms solving SSGs. The algorithm we propose relies on a refinement of this result, we will prove that optimal strategies can be looked for in a strict subset of positional strategies called the set of **f**-strategies.

In this section, we describe what are **f**-strategies.

3.1 Informal Description of f-Strategies

With every permutation $\mathbf{f} = (r_0, \ldots, r_m)$ of random vertices V_R, we associate a couple $\sigma_\mathbf{f}, \tau_\mathbf{f}$ of positional strategies, called the **f**-strategies. We give intuition about what are **f**-strategies, before giving their formal construction.

A permutation $\mathbf{f} = (r_0, \ldots, r_m)$ of V_R intuitively represents preferences of Max and Min over the random vertices: player Max prefers the play to start from a random vertex of index as high as possible, *i.e.* starting from vertex r_{l+1} is better for player Max than starting from vertex r_l, whereas the opposite holds for player Min.

When both players agree on the preferences given by **f**, the **f**-strategies $\sigma_\mathbf{f}, \tau_\mathbf{f}$ are natural behaviours of player Max and Min. Indeed, strategy $\sigma_\mathbf{f}$ for player Max consists in attracting the pebble either in the target vertex or in a random vertex of index as high as possible in **f**. By opposite, strategy $\tau_\mathbf{f}$ for player Min consists in keeping the pebble away from the target and from random vertices of high index in **f**.

The **f**-strategies can be described more precisely, introducing a non-stochastic game. In this non-stochastic game, plays can be either of finite or infinite duration, and after a play, players Max and Min have to give coins to each other. This game is played on the game graph $(V, V_{\text{Max}}, V_{\text{Min}}, E)$, where all random vertices are terminal vertices. When the play reaches a random vertex $r_l \in V_R$, the play immediately stops and player Min has to give l coins to Max, where l is the index of the random vertex r_l in the permutation $f = (r_0, \ldots, r_m)$. Of course player Min wants to give as few coins as possible to player Max, whereas the goal of player Max is the opposite. The worst for player Min is when the play does not reach a random vertex but reaches the target vertex instead, in that case player Min has to give $m + 1$ coins to Max. The best for player Min is the remaining case, when the play never reaches any random vertex nor the target,

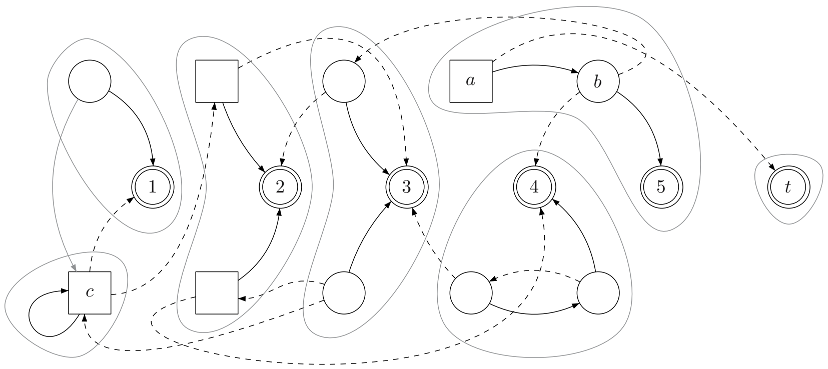

Fig. 2. The permutation is $\mathbf{f} = (1, 2, 3, 4, 5)$. Edges consistent with **f**-strategies are black, other edges are dotted. For example from vertex a, player Min prefers moving the pebble on vertex b than on the target vertex t. Indeed in the former case player Min has to pay 5 coins to Max whereas in the latter case he would have to pay 6 coins.

then instead of giving coins to Max player Min *receives* 1 coin from player Max. In this game there exist positional optimal strategies $\sigma_{\mathbf{f}}$ and $\tau_{\mathbf{f}}$ for both players. These strategies are precisely the **f**-strategies. This intuitive interpretation of **f**-strategies is depicted on Figure 2, for the permutation $(1, 2, 3, 4, 5)$.

In the rest of this section, we define formally the notion of **f**-strategies. For this we need to introduce in the next subsection the notion of deterministic attractor.

3.2 Deterministic Attractors

Let $W \subseteq V$ be a subset of vertices. The *deterministic attractor* in W is the set of vertices $\mathrm{Att}(W) \subseteq V$ from which Max has a strategy for attracting the play in W and avoiding at the same time any visit to a random vertex before the first visit to W. An *attraction strategy in* W is a positional strategy $\sigma^{\#}$ for Max which guarantees that every play starting from $\mathrm{Att}(W)$ has this property. A *trapping strategy out of* W is a positional strategy $\tau^{\#}$ for player Min which guarantees than any play starting outside $\mathrm{Att}(W)$ will avoid a visit to W before the first visit to a random vertex. These notions are formalized in the following proposition.

Proposition 1. *Let $W \subseteq V$ be a subset of vertices. There exists a subset* $\mathrm{Att}(W)$ *called the deterministic attractor in W, a positional strategy $\sigma^{\#}$ for* Max *called the attraction strategy in W and a positional strategy $\tau^{\#}$ for* Min *called the trapping strategy out of W such that:*

1. *For every $v_0 \in \mathrm{Att}(W)$, for every play $v_0 v_1 \cdots \in V^\omega$ consistent with $\sigma^{\#}$, there exists $n \in \mathbb{N}$ such that $v_n \in W$ and for every $0 \leq k < n, v_k \notin V_R$.*
2. *For every $v_0 \notin \mathrm{Att}(W)$, for every play $v_0 v_1 \cdots \in V^\omega$ consistent with $\tau^{\#}$, for every $n \in \mathbb{N}$, if $v_n \in \mathrm{Att}(W)$ then there exists $0 \leq k < n$ such that $v_k \in V_R$.*

There exists an algorithm that computes $\mathrm{Att}(W)$, $\sigma^{\#}$ *and* $\tau^{\#}$ *in time* $\mathcal{O}(|E|\cdot|V|)$.

3.3 Computing the f-Strategies

We now describe formally how to compute the **f**-strategies $\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}$ associated with a permutation $\mathbf{f} = (r_0, \ldots, r_m)$ of random vertices. Intuitively, the strategy $\sigma_{\mathbf{f}}$ for Max consists in attracting the play in the target vertex t or in a random vertex whose index is as high as possible in **f**, while the strategy $\tau_{\mathbf{f}}$ for Min aims at the opposite.

We start with defining a sequence $W_-, W_0, \ldots, W_{m+1}$ of subsets of V:

$$\begin{aligned} & W_{m+1} = \mathrm{Att}(\{t\}), \\ \text{for all } 0 \leq l \leq m, \quad & W_l = \mathrm{Att}(\{r_l, r_{l+1}, \ldots, r_m, t\}), \\ & W_- = V \setminus W_0. \end{aligned} \tag{6}$$

An example is given on Figure 2, where relative frontiers of the sets $W_-, W_0, \ldots, W_{m+1}$ are delimited by gray lines. On this example, the set W_{m+1}

only contains the target vertex t, the set W_5 is $\{a, b, 5, t\}$, the set W_- is the singleton $\{c\}$.

The $\mathbf{f}$-strategies are constructed by combining different positional strategies together. Let σ_{m+1} be the attraction strategy in $\{t\}$, then on W_{m+1} $\sigma_{\mathbf{f}}$ coincides with σ_{m+1} and $\tau_{\mathbf{f}}$ is any positional strategy. For $0 \leq l \leq m$, let σ_l be the attraction strategy in $\{r_l, r_{l+1}, \ldots, r_m, t\}$ and let τ_l be the trapping strategy out of $\{r_{l+1}, \ldots, t\}$, then on $W_l \setminus W_{l+1}$ $\sigma_{\mathbf{f}}$ coincides with σ_l and $\tau_{\mathbf{f}}$ coincides with τ_l. Let τ_- be the trapping strategy out of $\{r_0, \ldots, r_m, t\}$, then on $W_- = V \setminus W_0$, $\tau_{\mathbf{f}}$ coincides with τ_- and $\sigma_{\mathbf{f}}$ is any positional strategy.

We will use the following properties of $\mathbf{f}$-strategies:

Lemma 1. *Let* $\mathbf{f} = (r_0, \ldots, r_m)$ *a permutation of random vertices. Every play consistent with* $\sigma_{\mathbf{f}}$ *and starting from* W_{m+1} *stays in* W_{m+1} *and reaches the target vertex* t*. Every play consistent with* $\tau_{\mathbf{f}}$ *and starting from* W_- *stays in* W_- *and never reaches* t*. Let* $\phi : V \to \mathbb{R}$ *defined by* $\phi(v) = \mathbb{P}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}}(\mathrm{Reach}(t))$*. For every* $0 \leq l \leq m$*, for every* $v \in W_l \setminus W_{l+1}$*,* $\phi(v) = \phi(r_l)$*.*

4 Optimality of f-Strategies

A key property of $\mathbf{f}$-strategies is given in the following theorem.

Theorem 2. *In every SSG, there exists a permutation* $\mathbf{f}$ *of random vertices such that the* $\mathbf{f}$*-strategies* $\sigma_{\mathbf{f}}$ *and* $\tau_{\mathbf{f}}$ *are optimal.*

This theorem suggests the following algorithm for solving SSGs. It consists in testing, for each possible permutation $\mathbf{f}$ of random vertices, whether the $\mathbf{f}$-strategies are optimal. Since $\mathbf{f}$-strategies are positional, their optimality can be tested in polynomial time using linear programming [Der72, Con92]. Finally, the corresponding algorithm can find values and optimal strategies solving at most $|V_{\mathrm{R}}|!$ linear programs.

Testing optimality of $\mathbf{f}$-strategies can be done in a more elegant and efficient way, without making use of linear programming. Indeed, Theorem 3 shows that it is sufficient to test whether the $\mathbf{f}$-strategies are *self-consistent* and *progressive*, in the following sense.

Definition 1 (Self-consistent and progressive permutations). *Let* $\mathbf{f} = (r_0, \ldots, r_m)$ *be a permutation of random vertices and* $\sigma_{\mathbf{f}}$ *and* $\tau_{\mathbf{f}}$ *the* $\mathbf{f}$*-strategies. For* $v \in V$*, let* $\phi(v) = \mathbb{P}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}}(\mathrm{Reach}(t))$*.*

Permutation $\mathbf{f}$ *is* self-consistent *if*

$$\phi(r_0) \leq \phi(r_1) \leq \ldots \leq \phi(r_m). \tag{7}$$

Permutation $\mathbf{f}$ *is* progressive *if for every* $0 \leq i \leq j \leq m$*, if* $\phi(r_i) > 0$ *then there exists* $w \in \mathrm{Att}(\{r_{j+1}, \ldots, r_m, t\})$ *such that* $p(w|r_j) > 0$*.*

Both properties can be reformulated in term of the Markov chain induced by $\sigma_{\mathbf{f}}$ and $\tau_{\mathbf{f}}$, see Proposition 2.

Intuitively, both players use their **f**-strategies when they both agree on the preference order given by **f** and play consistently with this preference order. Self-consistency states that plays starting from random vertices of higher rank in **f** have greater probabilities of reaching the target. The progressive property states that if some random vertex r_i gives non-zero probability to reach the target, then, from every random vertex r_j of higher rank in **f**, also with non-zero probability either the target vertex or a random vertex of higher rank than r_j will be reached prior to any visit to a random vertex.

Next theorem states that together with self-consistency, the progressive property ensures optimality of the **f**-strategies.

Theorem 3. *Let* **f** *be a permutation of random vertices. If* **f** *is self-consistent and progressive then the* **f***-strategies* $\sigma_{\mathbf{f}}$ *and* $\tau_{\mathbf{f}}$ *are optimal. Moreover there exists a permutation* **f** *of random vertices which is self-consistent and progressive.*

Notice that self-consistency alone is not sufficient for ensuring that **f**-strategies are optimal, a counter-example is given on Figure 3.

The progressive property forces any wrong guess about preferences of the players to propagate among random vertices and to lead to a violation of self-consistency. Somehow, the progressive property plays a role similar to the halting hypothesis used in [Con92]. The halting hypothesis states that any play of the SSG should reach a sink vertex, which ensures uniqueness of a solution to the local optimality equations, see [Con92] for more details.

An algorithm for solving SSGs and based on Theorem 3 is described in the next section. The rest of this section is dedicated to the proof of Theorem 3, which relies on a few preliminary lemmas.

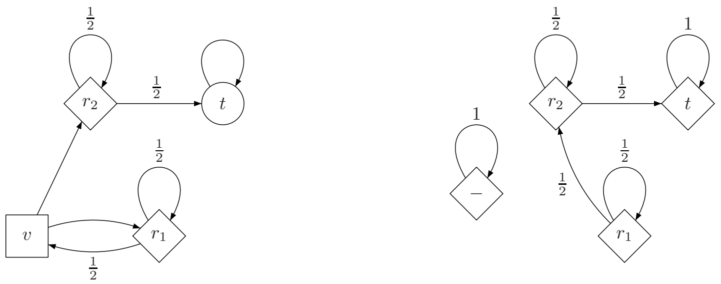

Fig. 3. On the left is depicted an SSG with two random vertices $\{r_1, r_2\}$, a Min vertex v and the target vertex t. Player Min has only one optimal strategy: moving the pebble to vertex r_1 whenever it is on v, this way the play never reaches the target vertex t. This is exactly the **f**-strategy $\tau_{\mathbf{f}}$ associated with the permutation $\mathbf{f} = (r_1, r_2)$. Suppose now the permutation is $\mathbf{f} = (r_2, r_1)$, then following its **f**-strategy $\tau_{\mathbf{f}}$, player Min will go to vertex r_2 from v. In that case, t is reached from both r_1 and r_2 with probability 1, hence permutation (r_2, r_1) is self-consistent, although strategy $\tau_{\mathbf{f}}$ is *not* optimal. However **f** is *not* progressive since from r_1 the play reaches r_2 before reaching t. On the right is depicted the Markov chain $\mathcal{M}_{\mathbf{f}}$ associated with $\mathbf{f} = (r_2, r_1)$.

Under the hypothesis that $\mathbf{f}$ is self-consistent, first lemma states that during a play consistent with $\sigma_{\mathbf{f}}$, the values of vertices relatively to $\mathbf{f}$-strategies is a super-martingale, and symmetrically for player Min.

Lemma 2. *Let* $\mathbf{f}$ *a permutation of* V_R *and* $\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}$ *the* $\mathbf{f}$*-strategies. Let* $\phi : V \to \mathbb{R}$ *defined by* $\phi(v) = \mathbb{P}_v^{\sigma_{\mathbf{f}},\tau_{\mathbf{f}}}(\text{Reach}(t))$. *Suppose* $\mathbf{f}$ *is self-consistent. Then for every strategies* σ, τ *for* Max *and* Min*, for every* $v \in V$ *and* $n \in \mathbb{N}$*,*

$$\mathbb{E}_v^{\sigma_{\mathbf{f}},\tau} [\ \phi(V_{n+1}) \ \mid V_0 \cdots V_n] \geq \phi(V_n), \tag{8}$$
$$\mathbb{E}_v^{\sigma,\tau_{\mathbf{f}}} [\ \phi(V_{n+1}) \ \mid V_0 \cdots V_n] \leq \phi(V_n). \tag{9}$$

Under the hypothesis that $\mathbf{f}$ is progressive, next lemma gives a necessary and sufficient condition for a play consistent with $\sigma_{\mathbf{f}}$ to reach the target vertex.

Lemma 3. *Let* $\mathbf{f}$ *be a permutation of* V_R *and* $\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}$ *the* $\mathbf{f}$*-strategies. Let* $\phi : V \to \mathbb{R}$ *defined by* $\phi(v) = \mathbb{P}_v^{\sigma_{\mathbf{f}},\tau_{\mathbf{f}}}(\text{Reach}(t))$. *Suppose* $\mathbf{f}$ *is progressive. Then for every vertex* $v \in V$ *and every strategy* τ *for* Min*:*

$$\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}\left(\text{Reach}(t) \,\middle|\, \phi(V_n) > 0 \textit{ for infinitely many } n \in \mathbb{N}\right) = 1. \tag{10}$$

Next lemma is the main ingredient for constructing iteratively a self-consistent and progressive permutation.

Lemma 4. *Let* $X \subseteq V$ *be a subset of vertices of an SSG and let* $W = \text{Att}(X)$. *Suppose* W *contains the target vertex. Then either all vertices* $v \in V \setminus W$ *have value* 0 *in the SSG or there exists a random vertex* $r \in V_R \cap (V \setminus W)$ *such that* $\text{val}(r) = \max\{\text{val}(v) \mid v \in V \setminus W\}$ *and* $\sum_{v \in W} p(v|r) > 0$.

We now give a proof of Theorem 3.

Proof (of Theorem 3)
We start with proving that if $\mathbf{f}$ is self-consistent and progressive then $\sigma_{\mathbf{f}}$ and $\tau_{\mathbf{f}}$ are optimal. Let $v \in V$ and σ, τ be some strategies for Max and Min.

We first prove that starting from v, $\sigma_{\mathbf{f}}$ ensures to reach t with probability at least $\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau_{\mathbf{f}}}(\text{Reach}(t))$:

$$\begin{aligned}\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}(\text{Reach}(t)) &\geq \mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}(\phi(V_n) > 0 \text{ for infinitely many } n \in \mathbb{N}) \\ &\geq \mathbb{E}_v^{\sigma_{\mathbf{f}},\tau}\left[\limsup_{n\in\mathbb{N}} \phi(V_n)\right] \geq \limsup_{n\in\mathbb{N}} \mathbb{E}_v^{\sigma_{\mathbf{f}},\tau}[\phi(V_n)] \\ &\geq \mathbb{E}_v^{\sigma_{\mathbf{f}},\tau}[\phi(V_0)] = \phi(v) = \mathbb{P}_v^{\sigma_{\mathbf{f}},\tau_{\mathbf{f}}}(\text{Reach}(t)),\end{aligned} \tag{11}$$

where the first inequality comes from Lemma 3, the second because values of ϕ are between 0 and 1, the third is a property of expectations, the fourth comes from Lemma 2 and the last two equalities hold because V_0 is equal to the starting vertex v and by definition of ϕ.

We now prove that starting from v, $\tau_{\mathbf{f}}$ ensures to reach t with probability no more than $\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau_{\mathbf{f}}}(\text{Reach}(t))$:

$$\begin{aligned}\mathbb{P}_v^{\sigma,\tau_{\mathbf{f}}}(\text{Reach}(t)) &\leq \mathbb{P}_v^{\sigma,\tau_{\mathbf{f}}}\left(\left(\liminf_{n\in\mathbb{N}} \phi(V_n)\right) = 1\right) \leq \mathbb{E}_v^{\sigma,\tau_{\mathbf{f}}}\left[\liminf_{n\in\mathbb{N}} \phi(V_n)\right] \\ &\leq \liminf_{n\in\mathbb{N}} \mathbb{E}_v^{\sigma,\tau_{\mathbf{f}}}[\phi(V_n)] \leq \mathbb{E}_v^{\sigma,\tau_{\mathbf{f}}}[\phi(V_0)] \\ &= \phi(v) = \mathbb{P}_v^{\sigma_{\mathbf{f}},\tau_{\mathbf{f}}}(\text{Reach}(t)), \end{aligned} \tag{12}$$

where the first inequality holds because t is an absorbing state and $\phi(t) = 1$, the second holds because values of ϕ are between 0 and 1, the third is a property of expectations, the fourth comes from Lemma 2 and the two last equalities hold because V_0 is equal to the starting vertex v and by definition of ϕ.

Finally, (11) and (12) together prove that $\sigma_{\mathbf{f}}$ and $\tau_{\mathbf{f}}$ are optimal.

We now prove the existence of a permutation $\mathbf{f}$ which is self-consistent and progressive. For a set W and a random vertex $r \in V_{\text{R}}$ we denote $p(W|r) = \sum_{w\in W} p(w|r)$ the probability of going to W from r.

We build a self-consistent and progressive permutation $\mathbf{f} = (r_0, r_1, \ldots, r_m)$ by iteration of the following iterative step.

Let $0 \leq l \leq m$, suppose that vertices $(r_{l+1}, \ldots, r_m)$ have already been chosen, let $X_{l+1} = \{r_{l+1}, \ldots, r_m, t\}$ and let $W_{l+1} = \text{Att}(X_{l+1})$. If $l > 0$ and all vertices in $V \setminus W_{l+1}$ have value 0, choose r_l to be any random vertex in $V_{\text{R}} \setminus X_{l+1}$. Otherwise, according to Lemma 4, there exists a random vertex r_l in $V \setminus W_{l+1}$ whose value is maximal in $V \setminus W_{l+1}$ and such that

$$p(W_{l+1}|r_l) > 0. \tag{13}$$

This achieves the inductive step.

Let $\mathbf{f} = (r_0, r_1, \ldots, r_m)$ be a permutation built according to this iterative procedure, we now prove that $\mathbf{f}$ is self-consistent and progressive.

By construction of $\mathbf{f}$,

$$\text{val}(r_0) \leq \ldots \leq \text{val}(r_n). \tag{14}$$

By definition of the $\mathbf{f}$-strategies, for any $0 \leq l \leq m$:

(A) $\sigma_{\mathbf{f}}$ coincides on $W_l \setminus W_{l+1}$ with an attraction strategy in $\{r_l, \ldots, r_m, t\}$,
(B) $\tau_{\mathbf{f}}$ coincides on $W_l \setminus W_{l+1}$ with a trapping strategy out of $\{r_{l+1}, \ldots, r_m, t\}$,
(C) $\sigma_{\mathbf{f}}$ coincides on W_{m+1} with an attraction strategy in $\{t\}$,
(D) $\tau_{\mathbf{f}}$ coincides on $W_{-} = V \setminus W_0$ with a trapping strategy out of W_0,
(E) any play consistent with $\tau_{\mathbf{f}}$ starting form a vertex of value 0 stays in the set of vertices of value 0.

We start with proving that $\mathbf{f}$ is progressive. Let $0 \leq k \leq m$ such that $\mathbb{P}_{r_k}^{\sigma_{\mathbf{f}},\tau_{\mathbf{f}}}(\text{Reach}(t)) > 0$. According to (E), $\text{val}(r_k) > 0$, hence according to (14), for every $k \leq l \leq m$, $\text{val}(r_l) > 0$. Hence, for every $k \leq l \leq m$, eq. (13) holds, which proves that $\mathbf{f}$ is progressive.

Now we prove that $\mathbf{f}$ is self-consistent. According to (14), for proving that $\mathbf{f}$ is self-consistent it is enough to prove that for any $0 \leq l \leq m$,

$$\mathbb{P}_{r_l}^{\sigma_{\mathbf{f}},\tau_{\mathbf{f}}}(\text{Reach}(t)) = \text{val}(r_l). \tag{15}$$

We start with proving for every $0 \leq l \leq m$,

$$\text{val} \text{ is constant equal to } \text{val}(r_l) \text{ on } W_l \setminus W_{l+1}. \tag{16}$$

Let $0 \leq l \leq m$ and $v \in W_l \setminus W_{l+1}$. According to (A), $\sigma_{\mathbf{f}}$ guarantees any play starting from v to reach set $\{r_l, \dots, r_m, t\}$ hence $\text{val}(v) \geq \min\{\text{val}(r_l), \dots, \text{val}(r_m), \text{val}(t)\}$, and together with (14) we get $\text{val}(v) \geq \text{val}(r_l)$. The converse inequality holds because according to (B), strategy $\tau_{\mathbf{f}}$ guarantees any play starting from v to either stay forever in $V \setminus W_{l+1}$ and never reach t or to reach a random vertex in $\{r_0, \dots, r_l\}$, hence $\text{val}(v) \leq \max\{\text{val}(r_0), \dots, \text{val}(r_l)\} = \text{val}(r_l)$. This achieves to prove (16).

Now we prove that for every $v, w \in V$,

$$\mathbb{E}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} [\text{val}(V_{n+1}) \mid V_n = w] = \text{val}(w). \tag{17}$$

According to (C), val is constant equal to 1 on W_{m+1} and W_{m+1} is stable under $\sigma_{\mathbf{f}}$ and $\tau_{\mathbf{f}}$ hence (17) holds for $w \in W_{m+1}$. According to (D), val is constant equal to 0 on W_- and W_- is stable under $\sigma_{\mathbf{f}}$ and $\tau_{\mathbf{f}}$ hence (17) holds for $w \in W_-$. Let $0 \leq l \leq m$ and $w \in W_l \setminus W_{l+1}$. According to (16), val is constant on $W_l \setminus W_{l+1}$ and according to (A) and (B), $W_l \setminus W_{l+1}$ is stable under $\sigma_{\mathbf{f}}$ and $\tau_{\mathbf{f}}$, hence (17) holds if $w \in V_{\text{Max}} \cup V_{\text{Min}}$. If $w \in V_{\text{R}}$ then (17) also holds because $w \neq t$ hence according to (5), $\text{val}(w) = \sum_{v \in V} p(v|w) \cdot \text{val}(v)$.

Now we prove that for any $v \in V$,

$$(\mathbb{P}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} (\text{Reach}(t)) = 0) \implies (\text{val}(v) = 0). \tag{18}$$

For every $v \in V$ let $\phi(v) = \mathbb{P}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} (\text{Reach}(t))$. Let $Z = \{v \in V \mid \phi(v) = 0\}$. We start with proving that $Z = V \setminus W_l$ for some l. According to Lemma 1, for each $0 \leq l \leq m$, the value of ϕ is constant equal to $\phi(w_l)$ on the set $W_l \setminus W_{l+1}$. Since ϕ has value 0 on $V \setminus W_0$, and since $W_0 \subseteq W_1 \subseteq \ldots \subseteq W_{m+1}$, there exists $0 \leq l \leq m$ such that $Z = V \setminus W_l$. Now we prove that Z is stable under random moves. Indeed, let $r \in V_{\text{R}} \cap Z$ then since $\phi(r) = \sum_{v \in V} p(v|r)\phi(v)$ and since $r \in Z$, $\phi(r) = 0$, hence all successors of r have value 0 and are in Z. Now we prove that $\tau_{\mathbf{f}}$ guarantees that every play starting from Z never leaves Z. Indeed according to (B), strategy $\tau_{\mathbf{f}}$ traps the play in $V \setminus W_{l+1} = Z$ until it reaches a random vertex, but Z is stable under random moves. Finally, since Z does not contain the target, any play consistent with strategy $\tau_{\mathbf{f}}$ and starting from Z will never reach the target. This proves that vertices in Z have value 0 and achieves the proof of (18).

Now we can achieve the proof that $\mathbf{f}$ is self-consistent. We already proved that $\mathbf{f}$ is progressive hence according to Lemma 3, there are two types of play consistent with $\sigma_{\mathbf{f}}$: those reaching the target and those staying ultimately in the set where ϕ has value 0. In the former case, since t is absorbing, $\lim_n \text{val}(V_n) = 1$ and in the latter case according to (18), $\lim_n \text{val}(V_n) = 0$. Hence:

$$\mathbb{P}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} (\text{Reach}(t)) = \mathbb{E}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} \left[\lim_n \text{val}(V_n)\right] = \lim_n \mathbb{E}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} [\text{val}(V_n)] = \text{val}(v),$$

where the second equality holds because val is bounded and the third comes from (17). This proves (15) and achieves the proof that $\mathbf{f}$ is self-consistent.

5 An Algorithm for Computing Values of SSGs

In this section, we give an algorithm that computes values and optimal strategies of an SSG. This algorithm, based on Theorem 3, looks for a permutation $\mathbf{f}$ which is self-consistent and progressive.

5.1 Testing Whether a Permutation Is Self-consistent and Progressive

For testing whether a permutation $\mathbf{f}$ is self-consistent and progressive, it is enough to compute some reachability probabilities in the following Markov chain.

Definition 2 (Markov chain associated with f). *Let $\mathbf{f} = (r_0, \ldots, r_m)$ a permutation of V_R, and $W_-, W_0, \ldots, W_{m+1}$ the subsets of V defined by* (6)*. Let $\mathcal{M}_\mathbf{f}$ be the Markov chain with states $S = \{-, 0, \ldots, m, m+1\}$ such that both states $-$ and $m+1$ are absorbing and for every $i \in \{0, \ldots, m\}$ and $j \in S$, the transition probability from i to j in $\mathcal{M}_\mathbf{f}$ is given by:*

$$x_{i,j} = \sum_{v \in W_j} p(v|r_i).$$

The Markov chain $\mathcal{M}_\mathbf{f}$ is designed to mimic behaviour of the play when the players use their $\mathbf{f}$-strategies: it is obtained by removing edges which are not consistent with $\mathbf{f}$-strategies and shrinking each set W_l to the vertex r_l.

The following proposition gives an effective procedure for testing self-consistency and progressiveness of a permutation.

Proposition 2. *Let $\mathbf{f} = (r_0, \ldots, r_m)$ be a permutation of random vertices, with $|V_R| = m+1$. Let $\mathcal{M}_\mathbf{f}$ the Markov chain associated with $\mathbf{f}$, with transition probabilities $(x_{i,j})_{i,j \in S}$. For $0 \le i \le m$, let x_i^* the probability of eventually reaching state $m+1$ starting from state i in the Markov chain $\mathcal{M}_\mathbf{f}$. Then $\mathbf{f}$ is* self-consistent *iff for every $0 \le i, j \le m$,*

$$(i \le j) \implies (x_i^* \le x_j^*), \tag{19}$$

and $\mathbf{f}$ is progressive *iff for every $0 \le i \le j \le m$,*

$$(x_i^* > 0) \implies (\text{there exists } j < k \le m+1 \text{ such that } x_{j,k} > 0). \tag{20}$$

Let $I \subseteq S$ the set of states from which $m+1$ is reachable in $\mathcal{M}_\mathbf{f}$ i.e. *such that $x_i^* > 0$. Then $(x_i^*)_{i \in I}$ is the unique solution of the following linear system:*

$$\begin{cases} x_i^* &= \sum_{j \in I} x_{i,j} \cdot x_j^*, \quad (i \in I \setminus \{m+1\}) \\ x_{m+1}^* &= 1. \end{cases} \tag{21}$$

Proof. Uniqueness of a solution of the linear system (21) is proved for example in [Con92], Lemma 1.

5.2 Solving SSGs in Time $\mathcal{O}(\ |V_{\mathrm{R}}|! \cdot (|V||E| + |p|)\ )$

Bringing together results about optimality of **f**-strategies and characterization of self-consistent and progressive permutations given by Proposition 2, we obtain an algorithm for solving SSG:

Theorem 4. *Values and optimal strategies of a simple stochastic game* $G = (V, V_{\mathrm{Max}}, V_{\mathrm{Min}}, V_R, E, t, p)$ *are computable in time* $\mathcal{O}(\ |V_R|! \cdot (|V||E| + |p|)\ )$, *where* $|p|$ *is the maximal bit-length of a transition probability in* p.

This algorithm enumerates all possible permutations $\mathbf{f}$ of V_{R}. For each permutation, the algorithm tests whether $\mathbf{f}$ is self-consistent and progressive. This is done by computing the sets $W_-, W_0, \ldots, W_m, W_{m+1}$ defined by (6), computing the transition probabilities $(x_{i,j})_{i,j \in S}$ of the Markov chain $\mathcal{M}_{\mathbf{f}}$ associated with $\mathbf{f}$, solving the linear system (21) and testing conditions (19) and (20). If the permutation fails the test then the algorithm proceeds to the next permutation. If the permutation passes the test, then the algorithm outputs the $\mathbf{f}$-strategies and the mapping $\mathrm{val} : V \to [0, 1]$ which associates 0 to the vertices in W_-, 1 to the vertices in W_{m+1} and x_l^* to the vertices in $W_l, 0 \leq l \leq m$.

Correctness of this algorithm comes from Theorem 3, which ensures the existence of a self-consistent and progressive permutation $\mathbf{f}$ and the optimality of $\mathbf{f}$-strategies associated with any such permutation. Proposition 2 validates the procedure used for testing self-consistency and progressiveness.

The complexity of the algorithm is $\mathcal{O}(\ |V_{\mathrm{R}}|! \cdot (|V||E| + |p|)\ )$. Indeed, there are exactly $|V_{\mathrm{R}}|!$ permutations of random vertices. For each permutation $\mathbf{f}$, the algorithm builds the Markov chain $\mathcal{M}_{\mathbf{f}}$. This is done by computing the deterministic attractors $W_{m+1}, \ldots, W_-$, which according to Proposition 1 takes time $\mathcal{O}(|E||V|)$. Then the algorithms solves the linear system (21), which can be done in time $|V_{\mathrm{R}}|^3 |p|$, see [Dix82]. The two tests (19) and (20) can be performed in time $\mathcal{O}(|V_{\mathrm{R}}|)$.

6 Conclusion

We presented an algorithm computing values and optimal strategies of an SSG in time $\mathcal{O}(\ |V_{\mathrm{R}}|! \cdot (|V||E| + |p|)\ )$. Our algorithm is particularly efficient for the SSGs with few random vertices and does not rely on quadratic or linear programming solvers.

A natural way of improving our algorithm would be to design a smart way of updating a permutation $\mathbf{f}$ in case it is not self-consistent or progressive, this way one would obtain a new kind of strategy improvement algorithm for solving SSGs.

References

[Con92] Condon, A.: The complexity of stochastic games. Information and Computation 96, 203–224 (1992)

[Con93] Condon, A.: On algorithms for simple stochastic games. In: Advances in computational complexity theory. DIMACS series in discrete mathematics and theoretical computer science, vol. 13, pp. 51–73 (1993)

[Der72] Derman, C.: Finite State Markov Decision Processes. Academic Press, London (1972)

[Dix82] Dixon, J.D.: Exact solution of linear equations using p-adic expansions. Numerische Mathematik 40, 137–141 (1982)

[GH07] Gimbert, H., Horn, F.: Solving simple stochastic games with few random vertices, `http://hal.archives-ouvertes.fr/hal-00195914/fr/`

[Hal07] Halman, N.: Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. Algorithmica 49, 37–50 (2007)

[Kac79] Kachiyan, L.G.: A polynomial time algorithm for linear programming. Soviet Math. Dokl. 20, 191–194 (1979)

[Lud95] Ludwig, W.: A subexponential randomized algorithm for the simple stochastic game problem. Information and Computation 117, 151–155 (1995)

[Ren88] Renegar, J.: A polynomial-time algorithm, based on newton's method, for linear programming. Mathematical Programming 40, 59–93 (1988)

[Sha53] Shapley, L.S.: Stochastic games. In: Proceedings of the National Academy of Science USA, vol. 39, pp. 1095–1100 (1953)

[Som05] Somla, R.: New algorithms for solving simple stochastic games. Electr. Notes Theor. Comput. Sci. 119(1), 51–65 (2005)