Solving Simple Stochastic Games with Few Random Vertices

Hugo Gimbert, Florian Horn

SOLVING SIMPLE STOCHASTIC GAMES WITH FEW RANDOM VERTICES

HUGO GIMBERT AND FLORIAN HORN

LaBRI, CNRS, Bordeaux, France
e-mail address: hugo.gimbert@labri.fr

CWI, Amsterdam, The Netherlands
e-mail address: f.horn@cwi.nl

Abstract. Simple stochastic games are two-player zero-sum stochastic games with turn-based moves, perfect information, and reachability winning conditions.

We present two new algorithms computing the values of simple stochastic games. Both of them rely on the existence of optimal *permutation strategies*, a class of positional strategies derived from permutations of the random vertices. The "permutation-enumeration" algorithm performs an exhaustive search among these strategies, while the "permutation-improvement" algorithm is based on successive improvements, *à la* Hoffman-Karp.

Our algorithms improve previously known algorithms in several aspects. First they run in polynomial time when the number of random vertices is fixed, so the problem of solving simple stochastic games is fixed-parameter tractable when the parameter is the number of random vertices. Furthermore, our algorithms do not require the input game to be transformed into a stopping game. Finally, the permutation-enumeration algorithm does not use linear programming, while the permutation-improvement algorithm may run in polynomial time.

Introduction

Simple stochastic games (SSGs) are played by two players called Max and Min in a sequence of steps. The players move a pebble along the edges of a directed graph (V, E) whose vertices are partionned into three sets: V_{Max}, V_{Min}, and V_{R}. 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 vertex of V_{R} (a *random* vertex), the outgoing edge is chosen randomly according to a fixed probability distribution. The players have opposite goals, as Max wants to reach a special sink vertex ◎ while Min wants to avoid it forever. An example of SSG is depicted in Figure 1, with vertices of V_{Max} represented as ○'s, vertices of V_{Min} represented as □'s, and vertices of V_{R} represented as △'s.

SSGs are a natural model of reactive systems. Consider, for example, a hardware component. It can be modelled as an SSG, whose vertices represent the global states of the component and the target is some error state to avoid. The nature of a given vertex

2000 ACM Subject Classification: Games, Stochastic Processes.

Key words and phrases: simple stochastic games, algorithm.

This research was partially supported by the french project ANR "DOTS". The second author held the tenure of an ERCIM "Alain Bensoussan" fellowship programme.

LOGICAL METHODS
IN COMPUTER SCIENCE

DOI:10.2168/LMCS-???

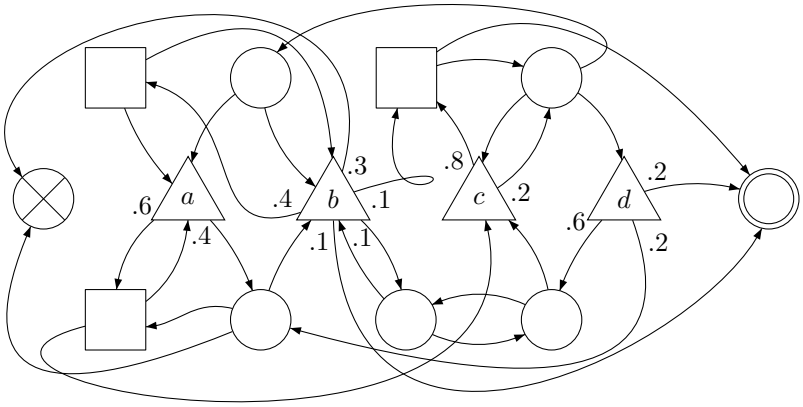

Figure 1: A Simple Stochastic Game.

depends on who can influence the immediate evolution of the system: it is a Min vertex if the software can choose between different options, a Max vertex if there is a (non-deterministic) input asked from the user, and a random vertex if the evolution depends on a stochastic environment. An optimal strategy for Min can then be used as the basis for the synthesis of a "good" driver, *i.e.* one which minimises the probability of entering the error state independently of the behaviour of the user.

The main algorithmic problem about SSGs is the computation of values of the vertices and optimal strategies for the players. This problem was first adressed by Condon, who showed that deciding whether the value of a vertex is greater than $\frac{1}{2}$ belongs to NP and co-NP [Con92]. Condon's algorithm guesses non-deterministically the values of vertices, which are rational numbers of linear size, and checks that they are solutions of some *local optimality equations*. This algorithm is correct only for *stopping* games, where the pebble reaches either the target or a sink target with probability one, regardless of the players' strategies. Any SSG can be transformed in polynomial time into a stopping SSG with (almost) the same values, but it incurs a quadratic blow-up of the size of the game.

Three other algorithms for solving SSGs are presented in [Con93]. The first one computes the values of the vertices using a quadratic program with linear constraints. The second one computes iteratively from below the values of the vertices, and the third is a strategy improvement algorithm *à la* Hoffman-Karp [HK66]. The two latter algorithms, as the ones recently proposed in [Som05], solve a series of linear programs which could be of exponential length. Furthermore, solving a linear program requires high-precision arithmetic, even if it can be done in polynomial time [Kha79, Ren88]. The best randomised algorithms achieve sub-exponential expected time $e^{O(\sqrt{n})}$ [Lud95, Hal07].

In this paper we present two algorithms computing the values and optimal strategies in SSGs: the "permutation-enumeration" and the "permutation-improvement" algorithms. The common basis for both algorithms is that optimal strategies can be looked for in a subset of the positional strategies called *permutation strategies*. Permutation strategies are derived from permutations over the random vertices. In order to find optimal strategies, the permutation-enumeration algorithm performs an exhaustive search among all permutation strategies, whereas the permutation-improvement algorithm performs successive improvements of permutation strategies, *à la* Hoffman-Karp [HK66].

The permutation-enumeration and the permutation-improvement algorithms share two advantages over existing algorithms. First, they perform much better on SSGs with few random vertices, as they run in polynomial time when the number of random vertices is logarithmic in the size of the game: it follows that the problem of solving SSGs is fixed-parameter tractable when the parameter is the number of random vertices. Second, they do not rely on the transformation of the input SSG into a stopping SSG, which avoids the quadratic blow-up of the size of the game. Moreover, the permutation-enumeration algorithm does not use linear or quadratic programming, (it just computes the solutions to linear systems) and its worst-case complexity is $\mathrm{O}\big(|V_{\mathrm{R}}|! \cdot (|E| + |\delta|)\big)$, where $|V_{\mathrm{R}}|$ is the number of random vertices, $|E|$ is the number of edges and $|\delta|$ is the maximal bit-length of transition probabilities. The nominal complexity of the permutation-improvement algorithm is higher but we do not know any non-trivial lower bound for its complexity: the permutation-improvement algorithm may actually run in polynomial time.

Outline. In Section 1, we provide formal definitions for SSGs, values and optimal strategies. We describe then in Section 2 the central notion of permutation strategies. Section 3 presents the permutation-enumeration algorithm, based on the *self-consistency* and *liveness* properties. Section 4 introduces an improvement policy for permutations which leads to the permutation-improvement algorithm.

1. Simple Stochastic Games

1.1. **Plays and strategies.** A *simple stochastic game* is a tuple $(V, V_{\mathrm{Max}}, V_{\mathrm{Min}}, V_{\mathrm{R}}, E, \delta, \circledcirc)$, where (V, E) is a graph, $(V_{\mathrm{Max}}, V_{\mathrm{Min}}, V_{\mathrm{R}})$ is a partition of V, and $\circledcirc$ is a distinguished sink vertex in V called the *target* of the game. The transitions from the random vertices are equipped with probabilities described by the function $\delta : V_{\mathrm{R}} \to V \to [0, 1]$, such that for all $v \in V_{\mathrm{R}}$, $w \in V$, $\delta(v)(w) > 0 \Rightarrow (v, w) \in E$, and $\sum_{w \in V} \delta(v)(w) = 1$.

An *infinite play* ρ is an infinite sequence $\rho_0\rho_1 \cdots \in V^\omega$ of vertices such that for all $i \in \mathbb{N}, (\rho_i, \rho_{i+1}) \in E$. It is *winning for* Max if there is a $i \in \mathbb{N}$ such that $\rho_i = \circledcirc$ (as $\circledcirc$ is a sink, it follows that $\forall j > i, \rho_j = \circledcirc$). Otherwise, ρ is *winning for* Min. A *finite play* is a finite prefix of an infinite play.

A (pure) *strategy* for Max is a mapping $\sigma : V^* V_{\mathrm{Max}} \to V$ such that for each finite play $h = h_0 \dots h_i$ ending in a Max vertex, $(h_i, \sigma(h)) \in E$. It is *positional* if it only depends on the last vertex of h: $\sigma(h) = \sigma(h_i)$. A play $\rho_0\rho_1 \dots$ is *consistent with* σ if for every i such that $\rho_i \in V_{\mathrm{Max}}$, $\rho_{i+1} = \sigma(\rho_0 \dots \rho_i)$. Strategies for Min are defined analogously and are generally denoted by τ.

1.2. **Measures and values.** The set of plays is made into a measurable space on the σ-algebra generated by the canonical projections $\{V_i\}_{i \in \mathbb{N}}$, where $V_i(\rho_0\rho_1 \dots) = \rho_i$ [Bil95]. Once an initial vertex v and two strategies σ and τ for players Max and Min have been fixed, the probability measure $\mathbb{P}_v^{\sigma,\tau}$ is defined by:

$$\begin{aligned}
&\mathbb{P}_v^{\sigma,\tau}\left(V_0 = v\right) = 1 \ , \\
&\mathbb{P}_v^{\sigma,\tau}\left(V_{i+1} = \sigma(V_0 \dots V_i) \mid V_i \in V_{\mathrm{Max}}\right) = 1 \ , \\
&\mathbb{P}_v^{\sigma,\tau}\left(V_{i+1} = \tau(V_0 \dots V_i) \mid V_i \in V_{\mathrm{Min}}\right) = 1 \ , \\
&\mathbb{P}_v^{\sigma,\tau}\left(V_{i+1} \mid V_i \in V_{\mathrm{R}}\right) = \delta(V_i)(V_{i+1}) \ .
\end{aligned}$$

The expectation of a real-valued, measurable and bounded function φ under $\mathbb{P}_v^{\sigma,\tau}$ is denoted $\mathbb{E}_v^{\sigma,\tau}[\varphi]$. We will often use implicitly the following formulae which rule the probabilities and expectations once a finite prefix $h = h_0 \dots h_i$ is fixed:

$$\mathbb{P}_v^{\sigma,\tau}(\Gamma \mid V_0 \dots V_i = h_0 \dots h_i) = \mathbb{P}_{h_i}^{\sigma[h],\tau[h]}(\Gamma[h]) \ , \tag{1.1}$$

$$\mathbb{E}_v^{\sigma,\tau}[\varphi \mid V_0 \dots V_i = h_0 \dots h_i] = \mathbb{E}_{h_i}^{\sigma[h],\tau[h]}[\varphi[h]] \ , \tag{1.2}$$

where $\sigma[h](\rho_0\rho_1\dots) = \sigma(h_0 \dots h_{i-1}\rho_0\rho_1\dots)$, and $\tau[h]$, $\Gamma[h]$, and $\varphi[h]$ are defined analogously.

If we fix only Max's strategy σ and the initial vertex v, the target vertex will be reached with probability at least:

$$\inf_\tau \mathbb{P}_v^{\sigma,\tau}(\mathrm{Reach}(\circledcirc)) \ ,$$

where $\mathrm{Reach}(\circledcirc)$ is the event $\{\exists i \in \mathbb{N}, V_i = \circledcirc\}$. Starting from v, player Max has strategies that guarantee a winning outcome with a probability greater than:

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

minus ϵ for any $\epsilon > 0$. Symmetrically, Min has strategies that guarantee a winning outcome with a probability less than:

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

plus ϵ for any $\epsilon > 0$. It is clear that $\mathrm{val}_*(v) \leq \mathrm{val}^*(v)$. In the case of SSGs, stronger results are known:

Theorem 1.1 ([Sha53, Gil57, LL69])**.** *Let $G = (V, V_{\mathrm{Max}}, V_{\mathrm{Min}}, V_R, E, \circledcirc, \delta)$ be a SSG. Then, for any vertex $v \in V$,*

$$\mathrm{val}_*(v) = \mathrm{val}^*(v) \ .$$

This common value is denoted by $\mathrm{val}(v)$. Furthermore, there are positional optimal strategies for both players, i.e. positional strategies $\sigma^\#$ and $\tau^\#$ such that, for any strategies σ and τ:

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

1.3. **Normalised games.** A SSG is *normalised* if the only vertex with value 1 is the target $\circledcirc$ and there is only one (sink) vertex $\otimes$ with value 0. Our motivations for the introduction of this notion are twofold. First, several proofs are much simpler for normalised games. Second, any SSG can be reduced to an equivalent normalised game in linear time and the resulting game is smaller than the original one. This reduction is presented on Figure 2: it simply consists in merging the region with value one into $\circledcirc$ and the region with value zero into a new sink vertex $\otimes$.

In the remainder of this article, we assume that we are working on a normalised SSG $G = (V, V_{\mathrm{Max}}, V_{\mathrm{Min}}, V_{\mathrm{R}}, E, \delta, \circledcirc, \otimes)$, with k random vertices.

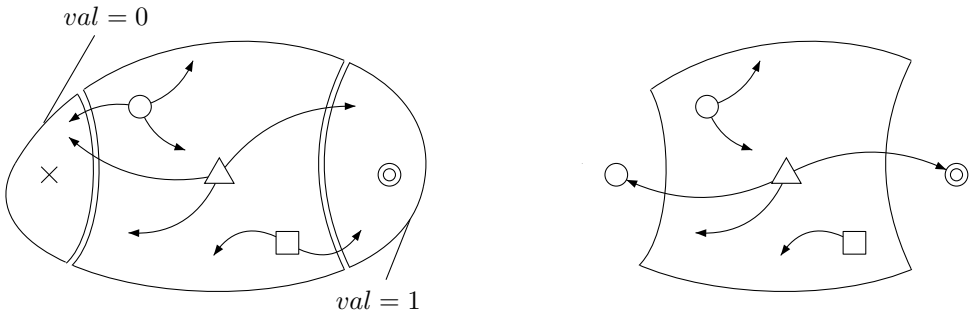

Figure 2: Normalisation.

2. Permutation strategies

The existence of positional optimal strategies is a key property of SSGs and the cornerstone of many algorithms solving these games. The algorithms we propose rely on a refinement of this result: optimal strategies can be looked for among a subset of the positional strategies, the set of "permutation strategies".

As a matter of fact, Theorem 1.1 is a corollary of results of the present paper. The proofs of our results often rely on the existence of values and optimal (not only ϵ-optimal) strategies in SSGs. This could be avoided —the main point is to use val_* instead of val— but we felt that it was not worth the extra complexity.

The main intuition underlying permutation strategies is that the only meaningful events in a play are the visits to random vertices. Between two visits the players only strive to impose which random vertex will be visited next, and the result of their interaction can be easily predicted. This is illustrated by Figure 3, which zooms on two details of Figure 1.

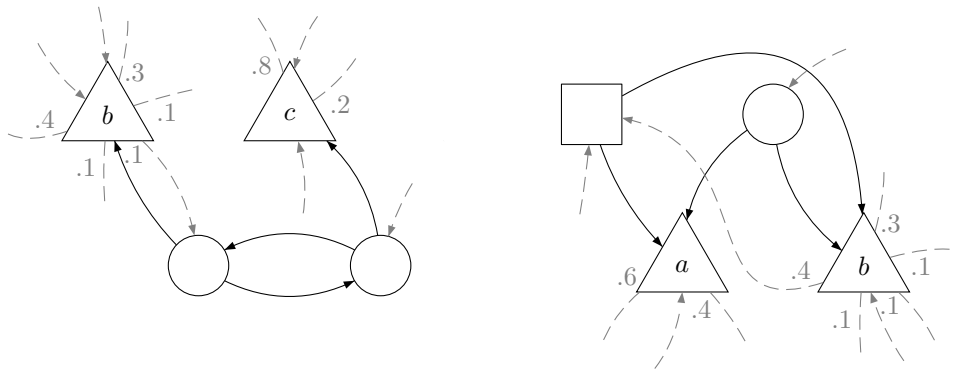

Figure 3: Coherence and contention.

In the left part of Figure 3, Max must choose between the two random vertices b and c (refusing to choose is not really an option). There is no reason to choose b in one of the vertices, and c in the other. We could consider only the strategies "always go to b" and "always go to c". In the right part of Figure 3, we consider relationships between the two

players' strategies. From their respective vertices $\bigcirc$ and $\Box$, Max and Min can send the pebble to either a or b. We could restrict our attention to the cases where Max goes to one, and Min to the other.

Underlying these intuitions is the idea of a "preference order" over the random vertices. In the remainder of this article, we formalise it as a *permutation*: a one-to-one correspondance $\mathbf{f}$ between V_{R} and $\{1,\dots,k\}$. For simplicity, we often write $\mathbf{f}_i$ instead of $\mathbf{f}^{-1}(i)$ and we consider the sink and target vertices as random vertices with the implicit assumption that they are respectively the lowest and greatest vertices: $\mathbf{f}_0 = \otimes$ and $\mathbf{f}_{k+1} = \circledcirc$.

2.1. **Attractors and f-regions.** Once a permutation $\mathbf{f} : V_{\mathrm{R}} \to \{1,\dots,k\}$ has been fixed, the $\mathbf{f}$-strategies consist in Max trying to reach the highest (with respect to $\mathbf{f}$) possible random vertex, while Min tries to thwart her. Notice that the situation is not exactly symmetric, since the burden of reaching a random vertex lies with Max: in case the pebble remains forever in controlled vertices then player Min wins. The formal definition of permutation strategies is based on the notion of *deterministic attractor*.

Definition 2.1. Let $X \subseteq V$ be a set of vertices. The deterministic attractor of Max to X, denoted by $\mathrm{DetAtt}(X)$, is computed recursively:

$$\begin{aligned} X^0 &= X \ , \\ X^{i+1} &= X^i \cup \left\{ v \in V_{\mathrm{Max}} \mid \exists w \in X^i, (v,w) \in E \right\} \\ &\qquad \cup \left\{ v \in V_{\mathrm{Min}} \mid \forall w \in V, (v,w) \in E \Rightarrow w \in X^i \right\} \ , \\ \mathrm{DetAtt}(X) &= \bigcup_{i>0} X^i \ . \end{aligned}$$

An attracting strategy to X for Max is a positional strategy σ such that:

$$\forall i \geq 1, \sigma(X^i) \subseteq X^{i-1} \ .$$

Symmetrically, a trapping strategy out of X for Min is a positional strategy τ such that:

$$\tau(V \setminus \mathrm{DetAtt}(X)) \subseteq V \setminus \mathrm{DetAtt}(X) \ .$$

The $\mathbf{f}$-*regions* associated with a permutation $\mathbf{f} : V_{\mathrm{R}} \to \{1,\dots,k\}$ are defined as embedded deterministic attractors to the random vertices:

$$\begin{aligned} W_{\mathbf{f}}[k+1] &= \{\circledcirc\} \ , \\ \forall 1 \leq i \leq k, W_{\mathbf{f}}[i] &= \mathrm{DetAtt}(\{\mathbf{f}_i,\dots,\mathbf{f}_k,\circledcirc\}) \setminus \bigcup_{j>i} W_{\mathbf{f}}[j] \ , \\ W_{\mathbf{f}}[0] &= \{\otimes\} \ . \end{aligned}$$

2.2. **Permutation strategies.** The $\mathbf{f}$-*strategies* $\sigma_{\mathbf{f}}$ and $\tau_{\mathbf{f}}$ are strategies such that, on each $W_{\mathbf{f}}[i]$:

- $\sigma_{\mathbf{f}}$ coincides with an attractor strategy to $\{\mathbf{f}_i,\dots,\mathbf{f}_k,\circledcirc\}$,
- $\tau_{\mathbf{f}}$ coincides with a trapping strategy out of $\{\mathbf{f}_{i+1},\dots,\mathbf{f}_k,\circledcirc\}$.

The $\mathbf{f}$-regions partition V, so we extend the definition domain of $\mathbf{f} : V_{\mathrm{R}} \to \{1, \dots, k\}$ to V in a natural way: $\mathbf{f}(v) = i$ if $v \in W_{\mathbf{f}}[i]$. The following properties are easy to prove:

$$\forall v \in V_{\mathrm{Max}}, \qquad \mathbf{f}(v) = \mathbf{f}(\sigma_{\mathbf{f}}(v)) \ , \tag{2.1}$$

$$\forall v \in V_{\mathrm{Max}}, \forall (v,w) \in E, \qquad \mathbf{f}(v) \geq \mathbf{f}(w) \ , \tag{2.2}$$

$$\forall v \in V_{\mathrm{Min}}, \qquad \mathbf{f}(v) = \mathbf{f}(\tau_{\mathbf{f}}(v)) \ , \tag{2.3}$$

$$\forall v \in V_{\mathrm{Min}}, \forall (v,w) \in E, \qquad \mathbf{f}(v) \leq \mathbf{f}(w) \ . \tag{2.4}$$

If Max plays $\sigma_{\mathbf{f}}$ and Min plays $\tau_{\mathbf{f}}$ from an initial vertex v, the first random vertex reached by the pebble is the unique random vertex w such that $\mathbf{f}(w) = \mathbf{f}(v)$. Figure 4 describes the $\mathbf{f}$-regions and $\mathbf{f}$-strategies of the game of Figure 1, for $\mathbf{f} = abcd$.

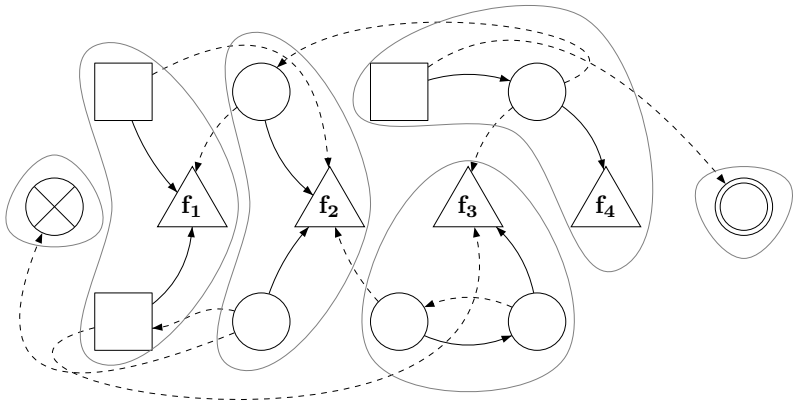

Figure 4: $\mathbf{f}$-regions and $\mathbf{f}$-strategies in the game of Figure 1.

2.3. **The f-values.** When both players use their respective permutation strategies, the probability that a pebble starting in v reaches $\circledcirc$ is denoted by $\varphi_{\mathbf{f}}(v)$:

$$\varphi_{\mathbf{f}}(v) = \mathbb{P}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}}(\mathrm{Reach}(\circledcirc)) \ .$$

Proposition 2.2. *Let $\mathbf{f}$ be a permutation. The $\mathbf{f}$-regions and the $\mathbf{f}$-strategies can be computed in time $\mathrm{O}\big(|E| \log^*(|V|)\big)$ and the $\mathbf{f}$-values can be computed in time $\mathrm{O}\big(|V_R|^3 |\delta|\big)$.*

Proof. The $\mathbf{f}$-regions and $\mathbf{f}$-strategies can be expressed in terms of *deterministic* games as they do not depend on what happens once a random vertex is reached. We can thus use the results of [AHMS08] to compute them in time $\mathcal{O}(|E| \log^*(|V|)$. In order to compute the $\mathbf{f}$-values, we build a Markov chain $\mathcal{M}_{\mathbf{f}}$ designed to mimic the behaviour of G when the players use their $\mathbf{f}$-strategies. Intuitively, we merge each region $W_{\mathbf{f}}[i]$ into a single vertex i; formally, $\mathcal{M}_{\mathbf{f}}$ is a Markov chain with states $S = \{0, \dots, k+1\}$ such that 0 and $k+1$ are absorbing and, for every $1 \leq i \leq k$ and $0 \leq j \leq k+1$, the transition probability from i to j is given by:

$$p_{ij} = \sum_{v \in W_{\mathbf{f}}[j]} \delta(\mathbf{f}_i)(v) \ .$$

The values $x^* : \{0, \dots, k+1\} \to [0,1]$ of $\mathcal{M}_{\mathbf{f}}$ are computed as follows. Let $I \subseteq S$ be the set of vertices from which $k+1$ is reachable in $\mathcal{M}_{\mathbf{f}}$. Then, for each $i \notin I$, $x_i^* = 0$, and

$(x_i^*)_{i\in I}$ is the unique solution of the following linear system:

$$\begin{cases} x_{k+1}^* & = 1 \\ x_i^* & = \sum_{j\in I} p_{i,j}\cdot x_j^* \ , \end{cases}$$

which can be solved in time $\mathrm{O}\big(|V_{\mathrm{R}}|^3|\delta|\big)$ [Dix82]. For each $v\in V$, $\varphi_{\mathbf{f}}(v) = x^*_{\mathbf{f}(v)}$. □

3. The permutation-enumeration algorithm

In this section we describe the permutation-enumeration algorithm which computes optimal strategies for both players. This algorithm relies on the following key property of permutation strategies.

Theorem 3.1. *In every SSG, there exists a permutation* $\mathbf{f}$ *such that* $\sigma_{\mathbf{f}}$ *is optimal for* Max *and* $\tau_{\mathbf{f}}$ *is optimal for* Min.

This theorem suggests a very simple enumerative algorithm computing values and optimal strategies: check for each permutation $\mathbf{f}$ whether the $\mathbf{f}$-strategies are optimal. Each test can be performed in polynomial time using linear programming [Der72, Con92]. However, linear programming requires high-precision arithmetic and is expensive in practice. Our permutation-enumeration algorithm uses a simpler criterion based on a refinement of Theorem 3.1: we look for permutations which are *live* and *self-consistent*.

3.1. Liveness and self-consistency.

The permutation-enumeration algorithm is based on two simple properties on permutations: self-consistency and liveness. Self-consistency expresses the adequation between *a priori* preferences (permutation $\mathbf{f}$) and resulting values (the $\mathbf{f}$-values $\varphi_{\mathbf{f}}$). Liveness stipulates that each random vertex has a positive probability to immediately lead to a better —from Max's point of view— region.

Definition 3.2. A permutation $\mathbf{f}$ is self-consistent if:

$$\varphi_{\mathbf{f}}(\mathbf{f}_1) \leq \varphi_{\mathbf{f}}(\mathbf{f}_2) \leq \ldots \leq \varphi_{\mathbf{f}}(\mathbf{f}_k) \ .$$

Definition 3.3. A permutation $\mathbf{f}$ is live if:

$$\forall 1\leq i\leq k, \exists j>i, \exists v\in W_{\mathbf{f}}[j], \delta(\mathbf{f}_i)(v) > 0 \ .$$

As we show below, the $\mathbf{f}$-strategies associated with a live and self-consistent permutation $\mathbf{f}$ are optimal and there is always such a permutation. The permutation-enumeration algorithm performs an exhaustive search for a live and self-consistent permutation.

Input: A normalised simple stochastic game $G = (V, V_{\mathrm{Max}}, V_{\mathrm{Min}}, V_{\mathrm{R}}, E, \delta, \circledcirc, \otimes)$.
Output: Optimal strategies for Max and Min.

```
forall permutation f over V_R do
    compute the f-regions;
    compute the f-values;
    if f is live and self-consistent then
        return σ_f and τ_f;
```

Algorithm 1: The permutation-enumeration algorithm.

Theorem 3.4. *The permutation-enumeration algorithm terminates and returns optimal strategies for* Max *and* Min. *Its worst-case running time is* $\mathrm{O}\big(|V_R|! \cdot (|E| + |\delta|)\big)$.

Proof. Correctness and termination are proved in Lemmas 3.7 and 3.10, respectively. The worst-case complexity follows from the fact that there are at most $k!$ permutations and Proposition 2.2. □

Before we proceed with the proofs of the main lemmas, let us make a case for liveness: Figure 5 shows that self-consistency is not enough to guarantee the optimality of the resulting strategies[1].

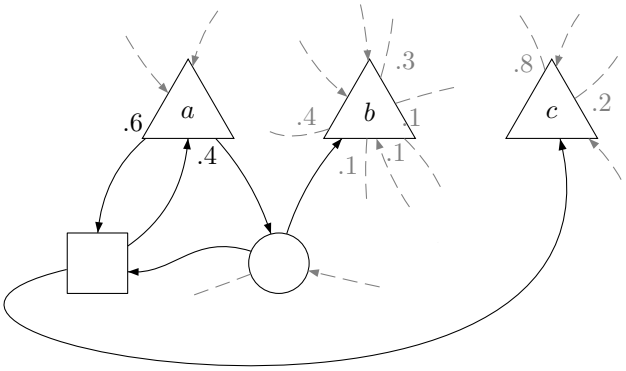

Figure 5: Self-consistency does not guarantee optimality.

In this excerpt from the game of Figure 1, Max's strategy in ○ should be to send the pebble to b, as Min could otherwise trap the play in $\{a, ○, □\}$. However, consider the permutation $\mathbf{g} = bcad$: Min sends the pebble from □ to c to avoid a; Max sends the pebble from ○ to □ to reach either a or c. We have thus $\varphi_{\mathbf{g}}(a) = \varphi_{\mathbf{g}}(c)$. As a matter of fact, we have $\varphi_{\mathbf{g}}(b) \leq \varphi_{\mathbf{g}}(a) = \varphi_{\mathbf{g}}(c) \leq \varphi_{\mathbf{g}}(d)$, so $\mathbf{g}$ is self-consistent even though the $\mathbf{g}$-values are not the correct ones. Liveness forbids this kind of gambits from Max. It replaces, in this aspect, the "stopping" hypothesis of Condon.

3.2. **Correctness of the permutation-enumeration algorithm.** We first show that if a permutation $\mathbf{f}$ is live and self-consistent, the $\mathbf{f}$-strategies are optimal (Lemma 3.7). We need two preliminary propositions. First, if $\mathbf{f}$ is self-consistent and Max plays according to $\sigma_{\mathbf{f}}$, the sequence $(\varphi_{\mathbf{f}}(V_i))_{i\in\mathbb{N}}$ is a submartingale[2] and symmetrically if $\mathbf{f}$ is self-consistent and Min plays according to $\tau_{\mathbf{f}}$ the sequence $(\varphi_{\mathbf{f}}(V_i))_{i\in\mathbb{N}}$ is a supermartingale.

Proposition 3.5. *Let* $\mathbf{f}$ *be a self-consistent permutation. Then, for any strategies* σ *and* τ *for* Max *and* Min, *vertex* v, *and integer* i,

$$\mathbb{E}_v^{\sigma_{\mathbf{f}},\tau}\left[\varphi_{\mathbf{f}}(V_{i+1}) \mid V_0 \dots V_i\right] \geq \varphi_{\mathbf{f}}(V_i) \ , \tag{3.1}$$

$$\mathbb{E}_v^{\sigma,\tau_{\mathbf{f}}}\left[\varphi_{\mathbf{f}}(V_{i+1}) \mid V_0 \dots V_i\right] \leq \varphi_{\mathbf{f}}(V_i) \ . \tag{3.2}$$

[1]It *would* be enough in stopping games, but testing liveness is cheaper than the reduction.

[2]We do not use any result about martingales in this paper.

Proof. In order to prove (3.1), it is enough to show that for any finite play $h = h_0 \dots h_i$,

$$\mathbb{E}_v^{\sigma_{\mathbf{f}},\tau}\left[\ \varphi_{\mathbf{f}}(V_{i+1})\ \mid V_0 \dots V_i = h_0 \dots h_i\right] \ \geq\ \varphi_{\mathbf{f}}(h_i)\ . \tag{3.3}$$

Depending on the owner of h_i, (3.3) follows from one of the following properties of $\varphi_{\mathbf{f}}$:

$$\forall v \in V_{\mathrm{Max}}, \qquad \varphi_{\mathbf{f}}(v) = \varphi_{\mathbf{f}}(\sigma_{\mathbf{f}}(v))\ , \tag{3.4}$$

$$\forall v \in V_{\mathrm{Min}}, \forall (v,w) \in E, \qquad \varphi_{\mathbf{f}}(v) \leq \varphi_{\mathbf{f}}(w)\ , \tag{3.5}$$

$$\forall v \in V_{\mathrm{R}}, \qquad \varphi_{\mathbf{f}}(v) = \sum_{w \in V} \delta(v)(w) \cdot \varphi_{\mathbf{f}}(w)\ . \tag{3.6}$$

The equations (3.4) and (3.6) follows from the definition of $\varphi_{\mathbf{f}}$, and (3.5) follows from the self-consistency of $\mathbf{f}$: by definition of the $\mathbf{f}$-regions, if $v \in V_{\mathrm{Min}}$ and $(v,w) \in E$ then $\mathbf{f}(v) \leq \mathbf{f}(w)$ (see (2.2)), so $\varphi_{\mathbf{f}}(v) \leq \varphi_{\mathbf{f}}(w)$. The proof of (3.2) is similar and we do not detail it. □

Second, we show a "stopping property" in the case where $\mathbf{f}$ is live and Max plays $\sigma_{\mathbf{f}}$.

Proposition 3.6. *Let* $\mathbf{f}$ *be a live permutation. Then, for any strategy* τ *for* Min *and initial vertex* v*,*

$$\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}\left(\mathrm{Reach}(\circledcirc) \vee \mathrm{Reach}(\otimes)\right) = 1\ .$$

Proof. By definition of liveness,

$$\alpha = \min_{1 \leq i \leq k} \sum_{w \in \bigcup_{j>i} W_{\mathbf{f}}[j]} \delta(\mathbf{f}_i)(w)$$

is positive. Let $n = |V|$ and $k = |V_{\mathrm{R}}|$ then the definition of $\sigma_{\mathbf{f}}$ yields:

$$\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}(V_n = \circledcirc \mid V_0 \neq \otimes) \geq \alpha^k\ ,$$

or, since $\circledcirc$ and $\otimes$ are sinks:

$$\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}(\forall m \leq n, V_m \notin \{\circledcirc, \otimes\}) \leq 1 - \alpha^k\ .$$

Equation (1.1) yields:

$$\forall i \in \mathbb{N}, \mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}(\forall m \leq i \cdot n, V_m \notin \{\circledcirc, \otimes\}) \leq (1-\alpha^k)^i\ ,$$

hence $\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}(\forall m \in \mathbb{N}, V_m \notin \{\circledcirc, \otimes\}) = 0$ hence Proposition 3.6. □

We can now prove the correctness of the permutation-enumeration algorithm:

Lemma 3.7. *Let* $\mathbf{f}$ *be a live and self-consistent permutation. Then* $\sigma_{\mathbf{f}}$ *is optimal for* Max *and* $\tau_{\mathbf{f}}$ *is optimal for* Min*.*

Proof. We first prove that $\sigma_{\mathbf{f}}$ ensures that a pebble starting from v has probability at least $\varphi_{\mathbf{f}}(v)$ to reach $\circledcirc$:

$$\begin{aligned}
\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}\left(\mathrm{Reach}(\circledcirc)\right) &= \mathbb{E}_v^{\sigma_{\mathbf{f}},\tau}\left[\lim_{i\in\mathbb{N}} \varphi_{\mathbf{f}}(V_i)\right] && (3.7)\\
&= \lim_{i\in\mathbb{N}} \mathbb{E}_v^{\sigma_{\mathbf{f}},\tau}\left[\varphi_{\mathbf{f}}(V_i)\right] && (3.8)\\
&\geq \mathbb{E}_v^{\sigma_{\mathbf{f}},\tau}\left[\varphi_{\mathbf{f}}(V_0)\right] = \varphi_{\mathbf{f}}(v)\ , && (3.9)
\end{aligned}$$

where (3.7) comes from Proposition 3.6, (3.8) is a property of expectations, and (3.9) comes from Proposition 3.5.

Then, we show that $\tau_{\mathbf{f}}$ ensures that a pebble starting from v has probability at most $\varphi_{\mathbf{f}}(v)$ to reach $\circledcirc$:

$$\begin{aligned} \mathbb{P}_v^{\sigma,\tau_{\mathbf{f}}}(\mathrm{Reach}(\circledcirc)) &\leq \mathbb{E}_v^{\sigma,\tau_{\mathbf{f}}}\left[\liminf_{i\in\mathbb{N}} \varphi_{\mathbf{f}}(V_i)\right] && (3.10)\\ &\leq \liminf_{i\in\mathbb{N}} \mathbb{E}_v^{\sigma,\tau_{\mathbf{f}}}[\varphi_{\mathbf{f}}(V_i)] && (3.11)\\ &\leq \mathbb{E}_v^{\sigma,\tau_{\mathbf{f}}}[\varphi_{\mathbf{f}}(V_0)] = \varphi_{\mathbf{f}}(v) \ , && (3.12) \end{aligned}$$

where (3.10) holds because $\circledcirc$ is a sink and $\varphi_{\mathbf{f}}(\circledcirc) = 1$, (3.11) is a property of expectations, and (3.12) comes from Proposition 3.5.

Thus, for any strategies σ and τ for Max and Min,

$$\mathbb{P}_v^{\sigma,\tau_{\mathbf{f}}}(\mathrm{Reach}(\circledcirc)) \leq \mathbb{P}_v^{\sigma_{\mathbf{f}},\tau_{\mathbf{f}}}(\mathrm{Reach}(\circledcirc)) \leq \mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}(\mathrm{Reach}(\circledcirc)) \ ,$$

which completes the proof of Lemma 3.7 □

3.3. **Termination of the permutation-enumeration algorithm.** Now we show the existence of a live and self-consistent permutation (Lemma 3.10). Our construction is based on Proposition 3.8 and its correctness on Proposition 3.9.

Proposition 3.8. *Let $X \subseteq V$ be a set of vertices including the target vertex $\circledcirc$ and Y be $V \setminus \mathrm{DetAtt}(X)$. Then either $Y = \{\otimes\}$ or there is a random vertex v in Y such that:*

$$\mathrm{val}(v) = \max\{\mathrm{val}(w) \mid w \in Y\} \ ,$$
$$\exists w \in \mathrm{DetAtt}(X), \delta(v)(w) > 0 \ .$$

Proof. Let Z be the set of vertices with maximal value in Y:

$$Z = \{v \in Y \mid \mathrm{val}(v) = \max_{w\in Y} \mathrm{val}(w)\} \ ,$$

and suppose that:

$$\forall v \in V_{\mathrm{R}} \cap Z, \forall w \in \mathrm{DetAtt}(X), \delta(v)(w) = 0 \ .$$

Let v be a vertex in Z. As G is normalised, we just need to show that $\mathrm{val}(v) = 0$, *i.e.* there is a strategy θ for Min such that for every strategy σ for Max, $\mathbb{P}_v^{\sigma,\theta}(\mathrm{Reach}(\circledcirc)) = 0$.

By definition of $\mathrm{DetAtt}(X)$, there is a positional strategy τ for Min such that $\tau(Y) \subseteq Y$, and it follows from the definition of Z that $\tau(Z) \subseteq Z$. As Z is also closed under random moves, a pebble starting in Z can only leave Z through a move of Max, which leads to $Y \setminus Z$ as $Y = V \setminus \mathrm{DetAtt}(X)$.

We define now a non-positional strategy θ in which Min plays according to τ as long as the play remains in Z and switches definitively to an optimal strategy the first time the pebble moves out of Z. We can thus partition the plays starting in v and consistent with σ and θ depending on if and where the play gets out of Z: Γ_Z is the set of plays remaining forever in Z, and for each w in $Y \setminus Z$, Γ_w is the set of plays where w is the first visited vertex outside of Z. Clearly $\mathbb{P}_v^{\sigma,\theta}(\mathrm{Reach}(\circledcirc) \mid \Gamma_Z) = 0$ and by definition of the strategy θ, $\forall w \in Y \setminus Z$, $\mathbb{P}_v^{\sigma,\theta}(\mathrm{Reach}(\circledcirc) \mid \Gamma_w) \leq \mathrm{val}(w)$. Hence $\mathbb{P}_v^{\sigma,\theta}(\mathrm{Reach}(\circledcirc)) \leq \max(0, \max_{w\in Y\setminus Z} \mathrm{val}(w))$ and since this holds for every σ, $\mathrm{val}(v) \leq \max(0, \max_{w\in Y\setminus Z} \mathrm{val}(w))$. By definition of Z this implies $\mathrm{val}(v) = 0$. □

Proposition 3.9. *Let* $\mathbf{f}$ *be a live permutation such that:*

$$\mathrm{val}(\mathbf{f}_1) \leq \mathrm{val}(\mathbf{f}_2) \leq \ldots \leq \mathrm{val}(\mathbf{f}_k) \ .$$

Then $\mathbf{f}$ *is self-consistent.*

Note that under the same hypotheses, Lemma 3.7 imply that $\mathbf{f}$-strategies are optimal.

Proof. We first show that:

$$\forall v \in V, \forall 1 \leq i \leq k, (\mathbf{f}(v) = i) \Rightarrow (\mathrm{val}(v) = \mathrm{val}(\mathbf{f}_i)) \ . \tag{3.13}$$

Consider the strategy σ^*, which mimics $\sigma_{\mathbf{f}}$ until the first time the pebble reaches a random vertex and then switches definitively to an optimal strategy. By definition of $\sigma_{\mathbf{f}}$, the first random vertex belongs to $\{\mathbf{f}_i, \ldots, \mathbf{f}_k, \circledcirc\}$, so σ^* ensures that a pebble starting in q reaches $\circledcirc$ with probability at least $\min\{\mathrm{val}(\mathbf{f}_i), \ldots, \mathrm{val}(\mathbf{f}_k), \mathrm{val}(\circledcirc)\} = \mathrm{val}(\mathbf{f}_i)$. A similar strategy τ^* for Min ensures that this probability is at most $\mathrm{val}(\mathbf{f}_i)$. So $\mathrm{val}(v) = \mathrm{val}(\mathbf{f}_i)$, and (3.13) follows.

Now we prove that val and $\varphi_{\mathbf{f}}$ coincide. According to (3.13) and by definition of permutation strategies,

$$\begin{aligned}
&\forall v \in V_{\mathrm{Max}}, && \mathrm{val}(v) = \mathrm{val}(\sigma_{\mathbf{f}}(v)) \ , \\
&\forall v \in V_{\mathrm{Min}}, && \mathrm{val}(v) = \mathrm{val}(\tau_{\mathbf{f}}(v)) \ , \\
&\forall v \in V_{\mathrm{R}}, && \mathrm{val}(v) = \sum_{w \in V} \delta(v)(w) \cdot \mathrm{val}(w) \ .
\end{aligned}$$

So, if Max and Min play according to their $\mathbf{f}$-strategies, the sequence $\mathrm{val}(V_i)_{i \in \mathbb{N}}$ is a martingale:

$$\mathbb{E}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} \left[\mathrm{val}(V_{i+1}) \ \middle| \ V_0 \ldots V_i \right] \ = \ \mathrm{val}(V_i) \ . \tag{3.14}$$

Consequenly, for every vertex v, $\varphi_{\mathbf{f}}(v) = \mathrm{val}(v)$:

$$\begin{aligned}
\varphi_{\mathbf{f}}(v) = \mathbb{P}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} (\mathrm{Reach}(\circledcirc)) \ &= \ \mathbb{E}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} \left[\lim_{i \in \mathbb{N}} \mathrm{val}(V_i) \right] && (3.15) \\
&= \ \lim_{i \in \mathbb{N}} \mathbb{E}_v^{\sigma_{\mathbf{f}}, \tau_{\mathbf{f}}} \left[\mathrm{val}(V_i) \right] && (3.16) \\
&= \ \mathbb{E}_v^{\sigma_{\mathbf{f}}, \tau} \left[\mathrm{val}(V_0) \right] = \mathrm{val}(v) \ , && (3.17)
\end{aligned}$$

where (3.15) comes from Proposition 3.6, (3.16) is a property of expectations, and (3.17) comes from (3.14). Since val and $\varphi_{\mathbf{f}}$ coincide, the hypothesis yields the self-consistency of $\mathbf{f}$. This completes the proof of Proposition 3.9. □

Lemma 3.10. *There exists a live and self-consistent permutation.*

Proof. We use iteratively Proposition 3.8 in order to build a permutation $\mathbf{f}$ such that, for every $k \geq i \geq 1$,

- $\mathrm{val}(\mathbf{f}_i) = \max \{\mathrm{val}(v) \mid v \in V \setminus \mathrm{DetAtt}(\mathbf{f}_{i+1}, \mathbf{f}_{i+2}, \ldots, \mathbf{f}_k)\}$;
- $\exists w \in \mathrm{DetAtt}(\mathbf{f}_{i+1}, \mathbf{f}_{i+2}, \ldots, \mathbf{f}_k), \delta(\mathbf{f}_i)(w) > 0$.

By construction $\mathbf{f}$ is live and $\mathrm{val}(\mathbf{f}_1) \leq \mathrm{val}(\mathbf{f}_2) \leq \ldots \leq \mathrm{val}(\mathbf{f}_k)$. Proposition 3.9 yields the self-consistency of $\mathbf{f}$, and Lemma 3.10 follows. □

4. The permutation-improvement algorithm

A drawback of the permutation-enumeration algorithm is that it considers each and every possible permutation of the random vertices, so $|V_{\mathrm{R}}|!$ is a lower bound for the worst-case complexity of this algorithm. Strategy-improvement algorithms avoid such enumerations, instead these algorithms proceed by successive improvements of a strategy: information about sub-optimality of a strategy is used to determine a "better" strategy, which ensures convergence to an optimal strategy. In this section, we emulate this idea with a permutation-improvement algorithm.

4.1. **A natural but incorrect improvement policy.** Starting from an initial permutation $\mathbf{f}$, we would like to improve $\mathbf{f}$ again and again until the permutation strategies $\sigma_{\mathbf{f}}$ and $\tau_{\mathbf{f}}$ are optimal. To test optimality we check that $\mathbf{f}$ is live and self-consistent (see Lemma 3.7). When $\mathbf{f}$ is live but *not* self-consistent we compute a new permutation $\mathbf{g}$ which is live and"better" than $\mathbf{f}$. A natural improvement policy consists in choosing $\mathbf{g}$ consistent with the $\mathbf{f}$-values i.e. $\mathbf{g}$ refines the pre-order induced by $\varphi_{\mathbf{f}}$. Unfortunately this is too naïve: the corresponding algorithm does not always terminate[3], a counter-example is given by Figure 6.

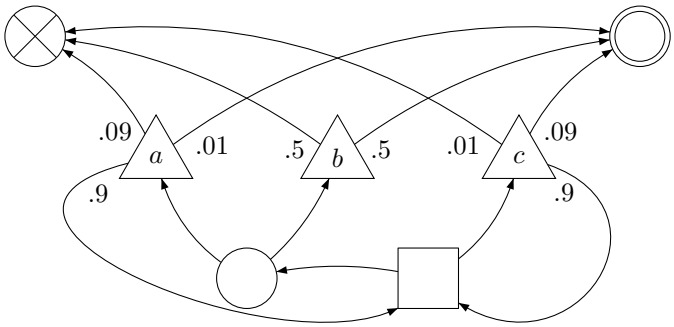

Figure 6: A counter-example for the naïve improvement algorithm.

If we start with the permutation $\mathbf{f} = acb$, the $\mathbf{f}$-strategies are as follows: in ○ Max goes to b and in □ Min goes to c. Hence, the $\mathbf{f}$-values of vertices a, c, and b are respectively .82, .9, and .5, so $\mathbf{f}$ is not self-consistent. The next permutation is $\mathbf{g} = bac$, and the following $\mathbf{g}$-strategies ensue: in ○ Max goes to a and in □ Min goes to ○. The $\mathbf{g}$-values of vertices b, a, and c are respectively .5, .1, and .18, so $\mathbf{g}$ is not self-consistent either. Moreover, the next permutation is $\mathbf{f} = acb$, so the naïve algorithm oscillates endlessly between $\mathbf{f}$ and $\mathbf{g}$, never reaching the correct permutation abc.

4.2. **A correct improvement policy.** The correct permutation-improvement policy is a bit more complex: given a live but not self-consistent permutation $\mathbf{f}$, we choose a permutation $\mathbf{g}$ which is live and self-consistent in the one-player game $G[\sigma_{\mathbf{f}}]$, where vertices of player Max have only one outgoing edge: the edge consistent with the positional $\mathbf{f}$-strategy $\sigma_{\mathbf{f}}$. This improvement policy guarantees that the value of $\sigma_{\mathbf{g}}$ is greater than the value of $\sigma_{\mathbf{f}}$ (see Lemma 4.4) and is implemented by the following algorithm.

[3]Actually, the naïve algorithm terminates (and is correct) in the special case of one-player games [Hor08].

Input: A normalised simple stochastic game $G = (V, V_{\text{Max}}, V_{\text{Min}}, V_{\text{R}}, E, \delta, \circledcirc, \otimes)$.
Output: Optimal strategies for Max and Min.
1 Pick a live permutation $\mathbf{f}$;
2 **repeat**
3 **if** $\mathbf{f}$ *is self-consistent in* G **then**
4 **return** $\sigma_{\mathbf{f}}$ *and* $\tau_{\mathbf{f}}$;
5 **else**
6 replace $\mathbf{f}$ with a live and self-consistent permutation in $G[\sigma_{\mathbf{f}}]$;

Algorithm 2: The permutation-improvement algorithm.

The computation of a live and self-consistent permutation in $G[\sigma_{\mathbf{f}}]$ in line 2 relies on the computation of values of the one-player game $G[\sigma_{\mathbf{f}}]$. Details are given in the proof of the following theorem.

Theorem 4.1. *The permutation-improvement algorithm terminates and returns optimal strategies for* Max *and* Min *in at most* $|V_R|!$ *improvement steps. Furthermore, each improvement step can be carried out in polynomial time.*

Proof. According to Lemma 4.2 the algorithm returns a permutation which is both live and self-consistent in G hence according to Lemma 3.7 the corresponding permutation strategies are optimal in G which proves correctness of the algorithm.

Termination and the maximal number of iterations follows from Lemma 4.4, which proves that sucessive strategies $\sigma_{\mathbf{f}}$ have better and better values.

The computation of a live and self-consistent permutation in $G[\sigma_{\mathbf{f}}]$ in line 2 is achieved in polynomial time in the following way. First, compute the values of the one-player game $G[\sigma_{\mathbf{f}}]$ using linear programming [HK79, Con93]. Second, build in linear time a live permutation $\mathbf{g}$ consistent with these values like in the proof of Lemma 3.10. The permutation $\mathbf{g}$ is such that $\text{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_1) \leq \text{val}\,\sigma_{\mathbf{f}}(\mathbf{g}_2) \leq \ldots \leq \text{val}\,\sigma_{\mathbf{f}}(\mathbf{g}_k)$, where $\text{val}_{\sigma_{\mathbf{f}}}$ denotes the values in the game $G[\sigma_{\mathbf{f}}]$. According to Proposition 4.3 the game $G[\sigma_{\mathbf{f}}]$ is normalised hence Proposition 3.9 guarantees that $\mathbf{g}$ is consistent in $G[\sigma_{\mathbf{f}}]$. □

Let us compare briefly the permutation-enumeration and the permutation-improvement algorithms. Each improvement step of the permutation-improvement algorithm requires the computation of values of a one-player SSG, which can be performed using linear programming. These values could be computed as well using a permutation-improvement policy or a strategy-improvement algorithm in order to avoid linear programming altogether. Either way, we have to forfeit one of the advantages of the permutation-enumeration algorithm: the computational simplicity of its inner loop. On the other hand, we do not know any non-trivial lower bound on the number of loops in a run of the permutation-improvement algorithm: it may be polynomial.

4.3. **Soundness and correctness of the permutation-improvement algorithm.** The correctness proof is based on the following two results.

Lemma 4.2. *Let* σ *be a positional strategy for* Max *and* $\mathbf{f}$ *be a permutation. If* $\mathbf{f}$ *is live in* $G[\sigma]$ *it is also live in* G.

Proof. Let $W_{\mathbf{f}}$ and $X_{\mathbf{f}}$ denote the $\mathbf{f}$-regions in G and $G[\sigma]$, respectively. By definition, $\cup_{j>i} W_{\mathbf{f}}[j]$ is the deterministic attractor of Max to $\{\mathbf{f}_i, \ldots, \mathbf{f}_k, \circledcirc\}$ in G, while $\cup_{j>i} X_{\mathbf{f}}[j]$ is the same attractor in $G[\sigma]$. As the moves of Max are restricted in $G[\sigma]$, we get

$$\forall 1 \leq i \leq k, \bigcup_{j>i} X_{\mathbf{f}}[j] \subseteq \bigcup_{j>i} W_{\mathbf{f}}[j] \ . \tag{4.1}$$

Thus, the liveness of $\mathbf{f}$ in G follows from its liveness in $G[\sigma]$, and Lemma 4.2 ensues. □

Proposition 4.3. *Let* $\mathbf{f}$ *be a live permutation. Then* $G[\sigma_{\mathbf{f}}]$ *is normalised.*

Proof. In the proof of Proposition 3.6, we have shown the existence of a positive real number α such that for any strategy τ for min and vertex $v \neq \otimes$, $\mathbb{P}_v^{\sigma_{\mathbf{f}},\tau}(V_n = \circledcirc) \geq \alpha^k$ hence only $\otimes$ has value 0 in $G[\sigma_{\mathbf{f}}]$. Clearly only $\circledcirc$ has value 1 in $G[\sigma_{\mathbf{f}}]$ hence Proposition 4.3 follows. □

4.4. **Termination of the permutation-improvement algorithm.** The value of a strategy σ is denoted val_σ and defined by:

$$\forall v \in V, \mathrm{val}_\sigma(v) = \inf_\tau \mathbb{P}_v^{\sigma,\tau}\left(\mathrm{Reach}(\circledcirc)\right) \ .$$

For proving termination of the permutation-improvement algorithm we prove that successive strategies $\sigma_{\mathbf{f}}$ chosen by the algorithm have greater and greater values.

Lemma 4.4. *Let* $\mathbf{f}$ *be a live permutation in* G *and* $\mathbf{g}$ *be a live and self-consistent permutation in* $G[\sigma_{\mathbf{f}}]$*. Then for all* $v \in V$*,*

$$\mathrm{val}_{\sigma_{\mathbf{f}}}(v) \leq \mathrm{val}_{\sigma_{\mathbf{g}}}(v) \ . \tag{4.2}$$

Moreover, if for all $v \in V$*,* $\mathrm{val}_{\sigma_{\mathbf{g}}}(v) = \mathrm{val}_{\sigma_{\mathbf{f}}}(v)$ *then* $\mathbf{g}$ *is self-consistent in* G*.*

Proof. A key remark in the proof of Lemma 4.4 is that:

$$\mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_1) \leq \mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_2) \leq \ldots \leq \mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_k) \ . \tag{4.3}$$

Let $\psi_{\mathbf{f},\mathbf{g}}$ be the $\mathbf{g}$-values in $G[\sigma_{\mathbf{f}}]$. The self-consistency of $\mathbf{g}$ in $G[\sigma_{\mathbf{f}}]$ is:

$$\psi_{\mathbf{f},\mathbf{g}}(\mathbf{g}_1) \leq \psi_{\mathbf{f},\mathbf{g}}(\mathbf{g}_2) \leq \ldots \leq \psi_{\mathbf{f},\mathbf{g}}(\mathbf{g}_k) \ .$$

Lemma 3.7 applied to $G[\sigma_{\mathbf{f}}]$ implies that the $\mathbf{g}$-strategy of player Min in $G[\sigma_{\mathbf{f}}]$ is optimal in $G[\sigma_{\mathbf{f}}]$ hence $\psi_{\mathbf{f},\mathbf{g}} = \mathrm{val}_{\sigma_{\mathbf{f}}}$ and (4.3) follows.

Consider now the sequence $(\sigma^n)_{n\in\mathbb{N}}$, where σ^n is the strategy where Max plays according to $\sigma_{\mathbf{g}}$ until the pebble has visited n random vertices, and plays according to $\sigma_{\mathbf{f}}$ afterwards. In particular $\sigma^0 = \sigma_{\mathbf{f}}$. We show that for every vertex v the sequence $(\mathrm{val}_{\sigma^n}(v))_{n\in\mathbb{N}}$ is non-decreasing and that its limit is less than $\mathrm{val}_{\sigma_{\mathbf{g}}}(v)$. Since $\sigma^0 = \sigma_{\mathbf{f}}$ this will prove Lemma (4.2).

We first show by induction that for any integer n,

$$\forall v \in V, \mathrm{val}_{\sigma^{n+1}}(v) \geq \mathrm{val}_{\sigma^n}(v) \ . \tag{4.4}$$

Basis ($n = 0$)*:* We have to prove that values of σ^1 are greater than values of $\sigma_{\mathbf{f}}$. Let v be a vertex, i be the index of the $\mathbf{g}$-region of v in G, and j be the index of the $\mathbf{g}$-region of v in $G[\sigma_{\mathbf{f}}]$. As the moves of Max are restricted in $G[\sigma_{\mathbf{f}}]$ the definition of the $\mathbf{g}$-regions gives $i \geq j$ and (4.3) yields:

$$\mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_i) \geq \mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_j) \ . \tag{4.5}$$

If Max plays with σ^1, the definition of $\sigma_{\mathbf{g}}$ ensures that the first random vertex belongs to $\{\mathbf{g}_i, \mathbf{g}_{i+1}, \dots, \mathbf{g}_k, \circledcirc\}$, so $\mathrm{val}_{\sigma^1}(v) \geq \min\{\mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_i), \mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_{i+1}), \dots, \mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_k), 1\}$ and (4.3) yields:

$$\mathrm{val}_{\sigma^1}(v) \geq \mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_i) \ . \tag{4.6}$$

On the other hand we prove:

$$\mathrm{val}_{\sigma_{\mathbf{f}}}(v) = \mathrm{val}_{\sigma_{\mathbf{f}}}(\mathbf{g}_j) \ . \tag{4.7}$$

Let $\psi_{\mathbf{f},\mathbf{g}}$ denote $\mathbf{g}$-values in $G[\sigma_{\mathbf{g}}]$. We have already proved above that $\psi_{\mathbf{f},\mathbf{g}}$ is equal to $\mathrm{val}_{\sigma_{\mathbf{f}}}$. By definition of j, $\mathbf{g}_j$ is the first random vertex in a play in $G[\sigma_{\mathbf{f}}]$ starting from v and consistent with a $\mathbf{g}$-strategy for Min in $G[\sigma_{\mathbf{f}}]$ hence $\psi_{\mathbf{f},\mathbf{g}}(v) = \psi_{\mathbf{f},\mathbf{g}}(\mathbf{g}_j)$ which yields (4.7).

It follows from (4.5), (4.6), and (4.7) that (4.4) holds for $n = 0$.

Inductive step ($n \Rightarrow n+1$): The strategies σ^{n+2} and σ^{n+1} coincides with $\sigma_{\mathbf{g}}$ until the first visit to a random vertex. Then σ^{n+2} switches to σ^{n+1} while σ^{n+1} switches to σ^n. By induction hypothesis, $\mathrm{val}_{\sigma^{n+1}} \geq \mathrm{val}_{\sigma^n}$, so $\mathrm{val}_{\sigma^{n+2}} \geq \mathrm{val}_{\sigma^{n+1}}$ and (4.4) holds for $n+1$.

Now we show that for every v, $\lim_{n\in\mathbb{N}} \mathrm{val}_{\sigma^n}(v) \leq \mathrm{val}_{\sigma_{\mathbf{g}}}(v)$. Let τ be a strategy for Min. We have:

$$\begin{aligned}
\mathbb{P}_v^{\sigma_{\mathbf{g}},\tau}(\mathrm{Reach}(\circledcirc)) &= \mathbb{P}_v^{\sigma_{\mathbf{g}},\tau}(\neg\,\mathrm{Reach}(\otimes)) \ , && (4.8)\\
&= \lim_n \mathbb{P}_v^{\sigma_{\mathbf{g}},\tau}(\forall m \leq n, V_m \neq \otimes) \ , \\
&= \lim_n \mathbb{P}_v^{\sigma^n,\tau}(\forall m \leq n, V_m \neq \otimes) \ , && (4.9)\\
&\geq \lim_n \mathbb{P}_v^{\sigma^n,\tau}(\mathrm{Reach}\circledcirc) \geq \lim_n \mathrm{val}_{\sigma^n}(v) \ , && (4.10)
\end{aligned}$$

where (4.8) follows from Proposition 3.6, (4.9) holds because $\sigma_{\mathbf{g}}$ coincides with σ^n for at least n steps, and (4.10) by event inclusion and by definition of the value. This holds for every strategy τ hence $\mathrm{val}_{\sigma_{\mathbf{g}}}(v) \geq \lim_n \mathrm{val}_{\sigma^n}(v)$.

Altogether, $\mathrm{val}_{\sigma_{\mathbf{f}}}(v) = \mathrm{val}_{\sigma^0}(v) \leq \mathrm{val}_{\sigma^1}(v) \leq \cdots \leq \lim_n \mathrm{val}_{\sigma^n}(v) \leq \mathrm{val}_{\sigma_{\mathbf{g}}}(v)$ hence (4.2), which achieves to prove the first part of the lemma.

Let us suppose now that $\mathrm{val}_{\sigma_{\mathbf{f}}} = \mathrm{val}_{\sigma_{\mathbf{g}}}$. Equation (4.3) yields:

$$\mathrm{val}_{\sigma_{\mathbf{g}}}(\mathbf{g}_1) \leq \mathrm{val}_{\sigma_{\mathbf{g}}}(\mathbf{g}_2) \leq \ldots \leq \mathrm{val}_{\sigma_{\mathbf{g}}}(\mathbf{g}_k) \ . \tag{4.11}$$

We can thus apply Proposition 3.9 to $\mathbf{g}$ in $G[\sigma_{\mathbf{g}}]$ which yields the self-consistency of $\mathbf{g}$ in $G[\sigma_{\mathbf{g}}]$. By definition of $\mathbf{g}$-zones, they coincide in G and $G[\sigma_{\mathbf{g}}]$ hence the $\mathbf{g}$-values are equal in G and $G[\sigma_{\mathbf{g}}]$ and $\mathbf{g}$ is also self-consistent in G. □

Conclusion

We have presented two algorithms computing optimal strategies in simple stochastic games: the permutation-enumeration and the permutation-improvement algorithms. Both of them rely on the existence of optimal permutation strategies. The permutation-enumeration algorithm simply tests every permutation until it finds a live and self-consistent one. The permutation-improvement algorithm uses a smarter policy in order to choose a "better" permutation in the next round, *à la* Hoffman-Karp.

The permutation-enumeration algorithm has exponential worst-case complexity but it is a witness that solving SSGs is fixed-parameter tractable when the parameter is the number of random vertices. The nominal complexity of the permutation-improvement algorithm is a bit higher but we do not know any non-trivial lower bound on the number of improvement steps: the permutation-improvement algorithm may actually run in polynomial time.

Whether simple stochastic games are solvable in polynomial time remains a challenging open question.

Acknowledgements We would like to thank Marcin Jurdziński for some fruitful discussions, the anonymous reviewers for several useful suggestions and Julien Cristau for his unvaluable comments during the writing of the final version.

References

[AHMS08] Daniel Andersson, Kristoffer Arnsfelt Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. Deterministic Graphical Games Revisited. In *Proceedings of CiE'08*, volume 5028 of *LNCS*, pages 1–10. Springer-Verlag, 2008.

[Bil95] Patrick Billingsley. *Probability and Measure*. John Wiley & Sons, 1995.

[Con92] Anne Condon. The Complexity of Stochastic Games. *Information and Computation*, 96(2):203–224, 1992.

[Con93] Anne Condon. On Algorithms for Simple Stochastic Games. In *Advances in Computational Complexity Theory*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 51–73. American Mathematical Society, 1993.

[Der72] Cyrus Derman. *Finite State Markovian Decision Processes*. Academic Press, 1972.

[Dix82] John D. Dixon. Exact Solution of Linear Equations Using p-adic Expansions. *Numerische Mathematik*, 40:137–141, 1982.

[Gil57] Dean Gillette. *Stochastic Games with Zero Stop Probability*, volume 3 of *Contributions to the Theory of Games*, pages 179–187. Princeton University Press, 1957.

[Hal07] Nir Halman. Simple Stochastic Games, Parity Games, Mean Payoff Games and Discounted Payoff Games are all LP-Type Problems. *Algorithmica*, 49:37–50, 2007.

[HK66] Alan J. Hoffman and Richard M. Karp. On Nonterminating Stochastic Games. *Management Science*, 12(5):359–370, 1966.

[HK79] Arie Hordijk and L.C.M. Kallenberg. Linear Programming and Markov Decision Chains. *Management Science*, 25(4):353–362, 1979.

[Hor08] Florian Horn. *Random Games*. PhD thesis, Université Paris 7 and RWTH Aachen, 2008.

[Kha79] Leonid G. Khachiyan. A Polynomial Algorithm in Linear Programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[LL69] Thomas M. Liggett and Steven A Lippman. Stochastic Games with Perfect Information and Time Average Payoff. *SIAM Review*, 11(4):604–607, 1969.

[Lud95] Walter Ludwig. A Subexponential Randomized Algorithm for the Simple Stochastic Game Problem. *Information and Computation*, 117(1):151–155, 1995.

[Ren88] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1):59–93, 1988.

[Sha53] Lloyd S. Shapley. Stochastic Games. In *Proceedings of the National Academy of Science of the USA*, volume 39, pages 1095–1100, 1953.

[Som05] Rafal Somla. New Algorithms for Solving Simple Stochastic Games. 119(1):51–65, 2005.