A Generic Strategy Improvement Method for Simple Stochastic Games

David Auger ✉
Université Paris Saclay, UVSQ, DAVID, France

Xavier Badin de Montjoye ✉
Université Paris Saclay, UVSQ, DAVID, France

Yann Strozecki ✉
Université Paris Saclay, UVSQ, DAVID, France

──── **Abstract** ────

We present a generic strategy improvement algorithm (GSIA) to find an optimal strategy of simple stochastic games (SSG). We prove the correctness of GSIA, and derive a general complexity bound, which implies and improves on the results of several articles. First, we remove the assumption that the SSG is stopping, which is usually obtained by a polynomial blowup of the game. Second, we prove a tight bound on the denominator of the values associated to a strategy, and use it to prove that all strategy improvement algorithms are in fact fixed parameter tractable in the number r of random vertices. All known strategy improvement algorithms can be seen as instances of GSIA, which allows to analyze the complexity of converge from below by Condon [14] and to propose a class of algorithms generalising Gimbert and Horn's algorithm [16, 17]. These algorithms terminate in at most $r!$ iterations, and for binary SSGs, they do less iterations than the current best deterministic algorithm given by Ibsen-Jensen and Miltersen [18].

1 Introduction

A *simple stochastic game*, or SSG, is a two-player turn-based zero-sum game with perfect information introduced by Condon [13]. It is a simpler version of *stochastic games*, previously defined by Shapley [23]. An SSG is played by two players MAX and MIN moving a pebble on a graph. Vertices are divided into MIN vertices, MAX vertices, random vertices and a target vertex for MAX. When the pebble reaches a MIN or MAX vertex, corresponding players move the pebble to a neighbouring vertex of their choice. If it reaches a random vertex, the next vertex is chosen at random following some probability law. Finally, when the pebble reaches the target vertex, MIN pays 1 to MAX . The goal of MIN is to minimise the probability to reach the target vertex while MAX must maximise this probability.

We study the algorithmic problem of *solving* an SSG, i.e. finding a pair of optimal strategies in an SSG, or equivalently the optimal value vector of the optimal probabilities for MAX to reach the sink from each vertex. There are always optimal strategies for both players that are positional [13], i.e. stationary and deterministic, but the number of positional strategies is exponential in the size of the game. Consequently, finding a pair of optimal strategies is a problem not known to be in FP, but it is in PPAD [20], a class included in FNP.

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).
Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 12; pp. 12:1–12:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Simple Stochastic Games can be used to simulate many classical games such as parity games, mean or discounted payoff games [2, 9]. Moreover, stochastic versions of these games are equivalent to SSGs [2], which underlines that SSGs are an important model to study. SSGs have applications in different domains such as model checking of modal μ-calculus [24], or modelling autonomous urban driving [11].

There are roughly three known methods to solve SSGs: strategy improvement, value iteration and quadratic programming. A strategy improvement algorithm (SIA) starts with a strategy for one player and improves it until it is optimal, whereas value iteration algorithms (VIA) update a value vector by elementary operations, which converges to the optimal value vector of the game. Implementations of those algorithms have been written and compared in [21].

Denote by n be the number of MAX vertices and r be the number of random vertices in an SSG. For SSGs with MAX vertices of outdegree 2, the best known deterministic algorithm is an SIA which makes at worst $O\left(2^n/n\right)$ iterations (see [25]), and the best known randomised algorithm is a SIA described by Ludwig in [22], which runs in $2^{O(\sqrt{n})}$.

Gimbert and Horn give an SIA in [16], running in $O^*\left(r!\right)$ iterations, namely a superpolynomial dependency in r only (O^* omits polynomial factors in r and n). For SSGs where random vertices have a probability distribution $(1/2, 1/2)$ (coin toss), Ibsen-Jensen and Miltersen present a VIA of complexity in $O^*\left(2^r\right)$ [18]. It turns out that all SIA runs in $O^*\left(2^r\right)$ on SSGs with probability distribution $(1/2, 1/2)$, as we prove in this article. The same complexity of $O^*\left(2^r\right)$ is obtained for general SSGs with a more involved randomised algorithm in [5].

Most of the aforementioned algorithms rely on the game being stopping, meaning that it structurally ends in a sink with probability 1. This condition is not restrictive since any SSG can be transformed into a stopping SSG while keeping the same optimal strategies. However, this transformation incurs a quadratic blow-up of the game and cannot be used in real life application. In this paper we give bounds in $O\left(2^r Poly(n)\right)$ computational time for some kinds of algorithm. Using the stopping transformation would have induced a $O\left(2^{nr^2} Poly(n)\right)$ complexity. The stopping restriction has been lifted for quadratic programming in [21] and before that for SIA and VIA in [10, 8].

Contributions

We introduce GSIA, a new meta-algorithm to solve SSGs in Sec. 3. This algorithm proves simultaneously the correctness of multiple algorithms ([14, 16, 15, 25, 18, 5]). In Sec. 4, we give a general complexity bound that matches or improves on previous bounds obtained by ad-hoc methods. We show that all these algorithms are fixed-parameter tractable in the number of random vertices. Moreover, we do not rely on the fact that the game is stopping, which was commonly used in the aforementioned papers. The proof of correctness relies on a notion of concatenation for strategies and an analysis of absorbing sets in the game, while the complexity bound is derived from a new and tight characterisation of the values of an SSG. Finally, in Sec. 5, we show how GSIA can be used to derive new algorithms, generalising classical ones. In particular, we exhibit a class of algorithms which generalise Gimbert and Horn's algorithm and use less iterations than Ibsen-Jensen and Miltersen's algorithm.

We emphasise that our goal here is not to define a new algorithm that would have a better –but still exponential– complexity bound than the state of the art (a sub-exponential algorithm for SSGs, like the ones found for parity games [6, 12], would already be a significant improvement), but rather to wrap-up a lot of previous research by showing that all known

SIA for SSGs, despite having emerged in different contexts and having ad-hoc proofs of convergence, are in fact instances of a general pattern that can be further expanded, and actually share the best known complexity bounds.

2 Simple Properties of Simple Stochastic Games

Some proofs are missing from this extended abstract for space reason, but they can be found in the appendices or in a long version [3].

2.1 Simple Stochastic Games

We give a generalised definition of *Simple Stochastic Game*, a two-player zero-sum game with turn-based moves and perfect information introduced by Anne Condon [14].

▶ **Definition 1.** *A Simple Stochastic Game (SSG) is a directed graph G, together with:*
1. *A partition of the vertex set V in four parts V_{MAX}, V_{MIN}, V_R and V_S (all possibly empty, except V_S), satisfying the following conditions:*
   a. *every vertex of V_{MAX}, V_{MIN} or V_R has at least one outgoing arc;*
   b. *every vertex of V_S has exactly one outgoing arc which is a loop on itself.*
2. *For every $x \in V_R$, a probability distribution $p_x(\cdot)$ with rational values, on the outneighbourhood of x.*
3. *For every $x \in V_S$, a value $\text{Val}(x)$ which is a rational number in the closed interval $[0,1]$.*

In the article, we denote $|V_{\text{MAX}}|$ by n and $|V_R|$ by r. Vertices from V_{MAX}, V_{MIN}, V_R and V_S are respectively called MAX vertices, MIN vertices, random vertices and sinks. For $x \in V$, we denote by $N^+(x)$ the set of outneighbours of x. We assume that for every $x \in V_R$ and $y \in V$, $y \in N^+(x)$ if and only if $p_x(y) > 0$.

The game is played as follows. The two players are named MAX and MIN. A token is positioned on a starting vertex x. If x is in V_{MAX} (resp. V_{MIN}) the MAX player (resp. the MIN player) chooses one of the outneighbours of x to move the token to. If x is in V_R, the token is randomly moved to one of the outneighbours of x according to the probability distribution $p_x(\cdot)$, independently of everything else. This process continues until the token reaches a sink s and then, player MIN has to pay $\text{Val}(s)$ to player MAX and the game stops. The problem we study is to *find the best possible strategies* for MIN and MAX, and the expected value that MIN has to pay to MAX while following those strategies.

We consider a slightly restricted class of SSGs where the probability distribution on each random vertex has a given precision and the value of the sinks are 0 and 1.

▶ **Definition 2.** *For q a positive integer, we say that an SSG is a q-SSG if there are only two sinks of value 0 and 1, and for all $x \in V_R$, there is an integer $q_x \leq q$ such that the probability distribution $p_x(\cdot)$ can be written as $p_x(x') = \dfrac{\ell_{x,x'}}{q_x}$ for all x' where $\ell_{x,x'}$ is a natural number.*

As an example, let x be a random vertex of a 2-SSG, and let $u \in N^+(x)$, then $p_x(u)$ can be equal to 0, 1/2 or 1. The case $p_x(u) = 0$ is forbidden by definition, and if $p_x(u) = 1$, then x is of degree one and can be removed (by redirecting arcs entering x directly to u), without changing anything about the outcome of the game. Hence, we suppose without loss of generality that each random vertex of a 2-SSG has degree 2 and has probability distribution $(1/2, 1/2)$. This definition matches the one of a *binary SSG*, given by Condon and used in most articles on SSGs, except that we allow here MAX and MIN vertices to have an outdegree larger than 2.

2.2   Play, History and Strategies

▶ **Definition 3.** *A* **play** *in G is an infinite sequence of vertices $X = (x_0, x_1, x_2, \cdots)$ such that for all $t \geq 0$, (x_t, x_{t+1}) is an arc of G.*

If for a play $X = (x_t)$ there is some $t \geq 0$ with $x_t = s \in V_S$, then all subsequent vertices in the play are also equal to s. In this case, we say that the play **reaches** sink vertex s and we define the **value of the play** $\mathrm{Val}(X)$ as $\mathrm{Val}(s)$. If the play reaches no sink, then we set $\mathrm{Val}(X) = 0$.

A **history** of G is a finite directed path $h = (x_0, x_1, \cdots, x_k)$. If the last vertex x_k is a MAX vertex (resp. MIN vertex), we say that h is a MAX history (resp. MIN history).

▶ **Definition 4.** *A* **general** MAX **strategy** *(resp. general* MIN *strategy) is a map σ assigning to every* MAX *history (resp.* MIN *history) $h = (x_0, x_1, \cdots, x_k)$ a vertex $\sigma(h)$ which is an outneighbour of x_k. The set of these strategies is denoted by $\Sigma_{gen}^{\mathrm{MAX}}$ (resp. $\Sigma_{gen}^{\mathrm{MIN}}$).*

For $\sigma \in \Sigma_{gen}^{\mathrm{MAX}}$ and $\tau \in \Sigma_{gen}^{\mathrm{MIN}}$, given a starting vertex x_0, we recursively define a random play $X = (X_0, X_1, \cdots)$ of G in the following way. At $t = 0$ let $X_0 = x_0$, and for $t \geq 0$:
- if $X_t \in V_{\mathrm{MAX}}$, define $X_{t+1} = \sigma(X_0, X_1, \cdots, X_t)$;
- if $X_t \in V_{\mathrm{MIN}}$, define $X_{t+1} = \tau(X_0, X_1, \cdots, X_t)$;
- if $X_t \in V_R$, then X_{t+1} is an outneighbour of X_t chosen following the probability distribution $p_{X_t}(\cdot)$, independently of everything else;
- if $X_t \in V_S$, define $X_{t+1} = X_t$.

This defines a distribution on plays which we denote by $\mathbb{P}_{\sigma,\tau}^{x_0}(\cdot)$, or simply $\mathbb{P}(\cdot)$ if strategies and starting vertex are clear from context. The corresponding expected value and conditional expected values are denoted by $\mathbb{E}_{\sigma,\tau}^{x_0}(\cdot|\cdot)$, or simply $\mathbb{E}(\cdot|\cdot)$.

We now define **positional strategies** which only depend on the last vertex in the history:

▶ **Definition 5.** *A general* MAX *strategy σ (resp.* MIN *strategy) is said to be positional if for any* MAX *vertex x (resp.* MIN *vertex) and any history $h = (x_0, \ldots, x)$, we have $\sigma(h) = \sigma((x))$ where (x) is the history containing only x as a start vertex. The set of positional* MAX *strategies (resp.* MIN *strategies) is denoted Σ^{MAX} (resp. Σ^{MIN}).*

2.3   Values in an SSG

▶ **Definition 6.** *Let G be an SSG and let (σ, τ) be a pair of* MAX *and* MIN *strategies, the value vector $v_{\sigma,\tau}^G$ is the real vector of dimension $|V|$ defined by, for any $x_0 \in V$,*

$$v_{\sigma,\tau}^G(x_0) = \mathbb{E}_{\sigma,\tau}^{x_0}\left(\mathit{Val}(X)\right).$$

This value represents the expected gains for player MAX *if both players plays according to (σ, τ) and the game starts in vertex x_0.*

As before, the superscript G can be omitted when the context is clear.

To compare value vectors, we use the pointwise order: we say that $v \geq v'$ if for all vertices $x \in V$ we have $v(x) \geq v'(x)$. Moreover, we say that $v > v'$ if $v \geq v'$ and there is some x such that $v(x) > v'(x)$. Given a MAX strategy σ, a **best response** to σ is a MIN strategy τ such that $v_{\sigma,\tau} \leq v_{\sigma,\tau'}$ for all MIN strategies τ'.

▶ **Proposition 7** ([14]). *A positional strategy admits a positional best response, which can be found in polynomial time using linear programming.*

The set of positional best responses to σ is denoted by $BR(\sigma)$. Similarly, for a MIN strategy τ, we define the notion of best response to τ and the corresponding set is denoted by $BR(\tau)$. Except explicitly stated otherwise (in Sec. 3.3), all considered strategies are positional.

We denote by $\tau(\sigma)$ a positional best response to σ. For a MAX strategy σ and $\tau \in BR(\sigma)$, we write v_σ for $v_{\sigma,\tau}$. For a MIN strategy τ and $\sigma \in BR(\tau)$, we write v_τ for $v_{\sigma,\tau}$. The vector v_σ is called the value vector of strategy σ, and is used to compare strategies by writing $\sigma' \underset{G}{>} \sigma$ if and only if $v_{\sigma'}^G > v_\sigma^G$.

It is well known (see [14, 25]) that there is a pair of deterministic positional strategies (σ^*, τ^*) called optimal strategies, that satisfies for all x, $v^* = v_{\sigma^*,\tau^*} = v_{\sigma^*} = v_{\tau^*}$ since σ^* and τ^* are best responses to each other.

2.4 Optimality Conditions

The next two lemmas give characterisations of (optimal) value vectors under a pair of strategies. They are fundamental to all algorithms finding optimal strategies. Proofs of similar results can be found in [13]; we add here a fifth condition to make the characterisation hold when the game is not stopping.

For any SSG, the vertices with value 0 under optimal strategies can be found in linear time by a simple graph traversal computing its complementary, the set of MAX vertices which can access a sink of positive value, regardless of the choice of the MIN player. Let K^G be the set of vertices with value 0 under optimal strategies. For a MAX strategy σ of G and a MIN strategy τ, we call $K_{\sigma,\tau}^G$ the set of vertices with value zero under the pair of strategies σ, τ, and K_σ^G the set of vertices with value zero under σ, τ when τ is a best response to σ.

▶ **Lemma 8.** *Given positional strategies (σ, τ) and a real $|V|$-dimensional vector v, one has equality between v and $v_{\sigma,\tau}$ if and only if the following conditions are met:*
 **(i)** *For $s \in V_S$, $v(s) = Val(s)$*
  (ii) *For $r \in V_R$, $v(r) = \displaystyle\sum_{y \in N^+(r)} p_r(y)v(y)$*
  (iii) *For $x \in V_{\text{MIN}}$, $v(x) = v(\tau(x))$*
  (iv) *For $x \in V_{\text{MAX}}$, $v(x) = v(\sigma(x))$*
  (v) *For any $x \in V$, $v(x) = 0$, if and only if $x \in K_{\sigma,\tau}^G$*

Moreover, $\tau \in BR(\sigma)$ if and only if for any x in V_{MIN}, $v(x) = \displaystyle\min_{y \in N^+(x)} v(y) = v(\tau(x))$ and the last condition is modified into $v(x) = 0$ if and only if $x \in K_\sigma^G$.

▶ **Lemma 9** (Optimality conditions). *Given positional strategies (σ, τ) and denoting $v = v_{\sigma,\tau}$, (σ, τ) are optimal strategies if and only if:*
  (i) *For $s \in V_S$, $v(s) = Val(s)$*
  (ii) *For $r \in V_R$, $v(r) = \displaystyle\sum_{y \in N^+(r)} p_r(y)v(y)$*
  (iii) *For $x \in V_{\text{MIN}}$, $v(x) = \displaystyle\min_{y \in N^+(x)} v(y)$*
  (iv) *For $x \in V_{\text{MAX}}$, $v(x) = \displaystyle\max_{y \in N^+(x)} v(y)$*
  (v) *For any $x \in V$, $v(x) = 0$, if and only if $x \in K^G$*

The conditions of Lemma 9 imply that (σ, τ) is a certificate of optimality that can be checked in polynomial time: compute $v_{\sigma,\tau}$ by solving the linear system of Lemma 8, compute K^G in linear time, then check in linear time if conditions are met.
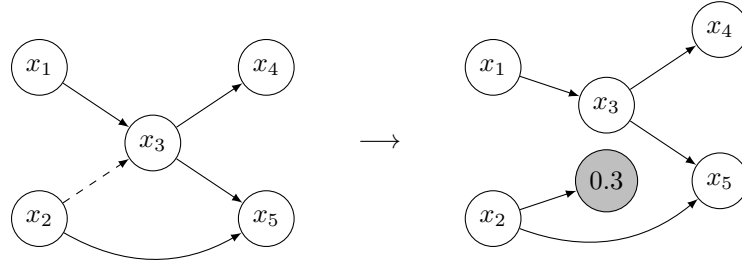
Figure 1 Transformation of the graph G in $G[\{(x_2, x_3)\}, f]$ where $f((x_2, x_3)) = 0.3$.

3 Generic Strategy Improvement Algorithm

3.1 Game Transformation

We present a simple transformation of an SSG, where some arcs of the game are rerouted to new sinks with appropriate values.

▶ **Definition 10.** *Let G be an SSG, A be a subset of the arcs of G and f be a function from A to the set of rational numbers. Let $G[A, f]$ be the SSG obtained from a copy of G with the following modifications: each arc $e = (x, y) \in A$ is removed and replaced in $G[A, f]$ by $e' = (x, s_e)$ where s_e is a new sink vertex with value $f(e)$. These new sinks of $G[A, f]$ are called A-sinks, and A is called the set of fixed arcs.*

Note that in the previous definition, the end vertex y of an arc $(x, y) \in A$ is not removed from the game. Its incoming arcs which are in A are simply redirected to sinks, see Fig. 1.

The function f is usually given by the values of a a strategy: we denote by $G[A, \sigma]$ the game $G[A, f]$, where f is defined on every arc $e = (x, y)$ of A by $f(e) = v_\sigma(y)$. Comparing G and $G[A, \sigma]$, the only differences are that arcs of A have their endpoints changed to new sinks. Therefore, a strategy defined in G can be interpreted as a strategy of $G[A, \sigma]$ and vice versa, and we identify strategies in G and $G[A, \sigma]$. However, when we compare the values of a strategy in both games (as in Lemma 12 below), it makes sense to compare only the values on vertices in G and not on A-sinks (and anyway values of A-sinks are fixed).

▶ **Lemma 11.** *For an SSG G, a subset of arcs A, and a MAX strategy σ, $K_\sigma^G = K_\sigma^{G[A,\sigma]}$.*

Proof. Fix a min strategy τ and define $R_{\sigma,\tau}^G(x)$ as the set of vertices that can be reached from x in G, following only arcs corresponding to σ and τ after MAX and MIN vertices, and any arc out of random vertices. We repeatedly use the easy fact that the three following assertions are equivalent:

(i) $v_{\sigma,\tau}^G(x) = 0$;
(ii) $v_{\sigma,\tau}^G(y) = 0$ for all $y \in R_{\sigma,\tau}^G(x)$;
(iii) $\mathrm{Val}^G(s) = 0$ for all $s \in V_S^G \cap R_{\sigma,\tau}^G(x)$.

The same equivalence is true in $G[A, \sigma]$, where we define $R_{\sigma,\tau}^{G[A,\sigma]}$ likewise. Denote by $R_A^G(x)$ vertices of $R_{\sigma,\tau}^G(x)$ that are endpoints of arcs in A, and let $S_A(x)$ be the corresponding A-sinks in $G[A, \sigma]$.

Suppose that $v_{\sigma,\tau}^G(x) = 0$ and consider a sink s in $V_S^{G[A,\sigma]} \cap R^{G[A,\sigma]}(x)$: either it belongs to V_S^G hence also to $R^G(x)$ and satisfies $\mathrm{Val}^G(s) = 0$ by (iii), or it belongs to $S_A(x)$ and then by definition

$$\mathrm{Val}^{G[A,\sigma]}(s) = v_\sigma^G(s) \leq v_{\sigma,\tau}^G(s) = 0.$$

Thus, by (iii) once again we have $v_{\sigma,\tau}^{G[A,\sigma]}(x) = 0$.

Conversely, suppose that $v_{\sigma,\tau}^{G[A,\sigma]}(x) = 0$ and let $s \in V_S^G \cap R_{\sigma,\tau}^G(x)$. Then, either $s \in R_{\sigma,\tau}^{G[A,\sigma]}$, hence by (iii)

$$\mathrm{Val}^G(s) = \mathrm{Val}^{G[A,\sigma]}(s) = 0,$$

or there is a $y \in R_A^G(x)$ such that $s \in R_{\sigma,\tau}^G(y)$. In this case we have $v_{\sigma,\tau}^G(y) = 0$ by (ii), hence $\mathrm{Val}^G(s) = 0$ by (iii) applied to y, and we see that $v_{\sigma,\tau}^G(x) = 0$.

Since we have $v_{\sigma,\tau}^G(x) = 0$ if and only if $v_{\sigma,\tau}^{G[A,\sigma]}(x) = 0$, regardless of τ, the result follows. ◀

▶ **Lemma 12.** *For an SSG G, a subset of arcs A, and a* MAX *strategy σ, $v_\sigma^G = v_\sigma^{G[A,\sigma]}$.*

Proof. This is a direct consequence of Lemma 8 and Lemma 11, since the vector v_σ^G satisfies the best-response conditions in $G[A,\sigma]$ and vice versa. ◀

3.2 The Algorithm

An SSG is **stopping** if under every pair of strategies, a play eventually reaches a sink with probability 1. Most algorithms in the literature depend on the game being stopping. It is usually not seen as a limitation since it is possible to transform every SSG into a stopping SSG, but the transformation makes the game polynomially larger by adding $O(nr)$ random vertices, which is bad from a complexity point of view, especially for algorithm with parametrized complexity in the number of random vertices. We strengthen the classical order on strategies, to get rid of the stopping condition in the generic strategy improvement algorithm presented in this section.

▶ **Definition 13.** *Let σ and σ' be two* MAX *strategies, then $\sigma' \underset{G}{\succ} \sigma$ if $\sigma' \underset{G}{>} \sigma$ and for every* MAX *vertex x, if $v_{\sigma'}^G(x) = v_\sigma^G(x)$, then $\sigma'(x) = \sigma(x)$.*

Algorithm 1 is a classical strategy improvement algorithm with two twists: the improvement is for *the stricter order $\succ$* and it is guaranteed in *the transformed game* rather than in the original game. We call Algorithm 1 the Generic Strategy Improvement Algorithm, or GSIA.

■ Algorithm 1 GSIA.

Data: G a stopping SSG
Result: (σ, τ) a pair of optimal strategies
1 begin
2   select an initial MAX strategy σ
3   **while** $(\sigma, \tau(\sigma))$ *are not optimal strategies of G* **do**
4   choose a subset A of arcs of G
5   find σ' such that $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$.
6 $\sigma \longleftarrow \sigma'$
7   **return** $(\sigma, \tau(\sigma))$

Algorithm 1 is a generic algorithm (or meta-algorithm) because neither the selection of an initial strategy σ at line 2, nor the way of choosing A at line 4, nor the way of finding σ' at line 5, are specified. A choice of implementation for these three parts is an **instance** of GSIA, that is a concrete strategy improvement algorithm.

Note that if $\sigma' \underset{G[A,\sigma]}{>} \sigma$ is found, it is easy to find σ'' with $\sigma'' \underset{G[A,\sigma]}{\succ} \sigma$: define σ'' as equal to σ', except for MAX vertices x such that $v_{\sigma'}^G(x) = v_\sigma^G(x)$ and $\sigma'(x) \neq \sigma(x)$ where $\sigma''(x)$ is defined as $\sigma(x)$.

When we prove some property of GSIA in this article, it means that the property is true for all instances of GSIA, that is regardless of the selection of the initial strategy, the set A and the method for selecting σ'.

In order to prove the correctness of GSIA, we need to prove two points:

1. If σ is not optimal in G, then σ is not optimal in $G[A,\sigma]$.
2. If $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$ then $\sigma' \underset{G}{\geq} \sigma$.

The first point is proved in the following lemma, while the second one is harder to obtain and is the subject of the next two subsections.

▶ **Lemma 14.** *For an SSG G and a subset of arcs A, a* MAX *strategy σ is optimal in G if and only if it is optimal in $G[A,\sigma]$.*

Proof. Except on A-sinks, the value vectors of σ in G and $G[A,\sigma]$ are equal by Lemma 12. Furthermore, by Lemma 11, $K_\sigma^G = K_\sigma^{G[A,\sigma]}$; hence σ satisfies the optimality conditions of Lemma 9 in G if and only if it satisfies them in $G[A,\sigma]$. ◀

3.3 Concatenation of Strategies

As a tool for proving the correctness of Algorithm 1, we introduce the notion of concatenation of strategies which produces non-positional strategies even if both concatenated strategies are positional. The idea of using a sequence of concatenated strategies to interpolate between two strategies has been introduced in [17].

▶ **Definition 15.** *For two* MAX *strategies σ, σ' and a subset of arcs A, we call $\sigma'|_A\sigma$ the non-positional strategy that plays like σ' until an arc of A is crossed, and then plays like σ until the end of the game. We let $\sigma'|_A^0\sigma = \sigma$ and for all $i \geq 0$, $\sigma'|_A^{i+1}\sigma = \sigma'|_A(\sigma'|_A^i\sigma)$.*

When A is clear from the context, we omit it and write $\sigma'|^i\sigma$. Strategy $\sigma'|_A^i\sigma$ is the strategy that plays like σ' until i arcs from A have been crossed, and then plays like σ. Hence, we can relate the strategy $\sigma'|_A\sigma$ to a positional strategy in $G[A,\sigma]$ as shown in the next lemma.

▶ **Lemma 16.** *For two* MAX *strategies σ, σ' and a subset of arcs A, we have: $v_{\sigma'|_A\sigma}^G = v_{\sigma'}^{G[A,\sigma]}$*

Proof. In G, after crossing an arc from A, by definition of $\sigma'|_A\sigma$, MAX plays according to σ. The game being memoryless, from this point, the best response for MIN is to play like $\tau(\sigma) \in BR(\sigma)$. Thus, there is a best response to $\sigma'|\sigma$ of the form $\tau'|\tau(\sigma)$ with τ' a MIN strategy not necessarily positional. Let us consider a play following $(\sigma'|\sigma, \tau|\tau(\sigma))$ with τ any MIN strategy. If the play does not cross an arc of A, then there is no difference between this play and a play following (σ', τ) in $G[A,\sigma]$. If an arc of A is used, then by Lemma 12 there is no difference between stopping with the value of $G[A,\sigma]$ or continuing in G while following (σ, τ). Thus we have: $v_{\sigma'|\sigma,\tau|\tau(\sigma)}^G = v_{\sigma',\tau}^{G[A,\sigma]}$.

Thus, if τ' is a best response to σ' in $G[A,\sigma]$, then $\tau'|\tau(\sigma)$ is a best response to $\sigma'|\sigma$ in G. This implies that $v_{\sigma'|\sigma}^G = v_{\sigma'}^{G[A,\sigma]}$. ◀

We now prove the fact that increasing the values of sinks can only increase the value of the game (a similar lemma is proved in [4]).

▶ **Lemma 17.** *Let G and G' be two identical SSGs except the values of theirs sinks $s \in V_S$, denoted respectively by $Val(s)$ and $Val'(s)$. If for every $s \in V_S$, $Val'(s) \geq Val(s)$, then for every MAX strategy σ we have $v_\sigma^{G'} \geq v_\sigma^G$.*

Proof. For $s \in V_S$, let $\mathbb{P}_{\sigma,\tau}^x(\to s)$ be the probability that the play ends in sink s while starting from vertex x, following strategies (σ, τ). For any vertex x we have:

$$v_{\sigma,\tau}^{G'}(x) = \sum_{s \in V_S'} \mathbb{P}_{\sigma,\tau}^x(\to s)Val'(s) \geq \sum_{s \in V_S'} \mathbb{P}_{\sigma,\tau}^x(\to s)Val(s) = v_{\sigma,\tau}^G(x)$$

This is true for any MIN strategy τ, thus $v_\sigma^{G'} \geq v_\sigma^G$. ◀

The following proposition is the core idea of GSIA: a strategy which improves on σ in the transformed game also improves on σ in the original game. The proof relies on a precise analysis of the set of vertices which cannot reach a sink, to deal with the fact that the game is not stopping. We prove that, if $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$ the limit of $v_{\sigma'|^i\sigma}^G$ is $v_{\sigma'}^G$ and the two previous lemmas imply $\sigma'|^i\sigma \geq \sigma'|^{i-1}\sigma > \sigma$, which yields the following proposition.

▶ **Proposition 18.** *Let G be an SSG, A a subset of arcs of G and σ, σ' two MAX strategies. If $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$ then $\sigma' \underset{G}{>} \sigma$.*

In order to avoid requiring the game to be stopping, it is necessary to pay particular attention to the set of vertices where the play can loop infinitely and yield value zero, which is a subset of the set of vertices of value 0. We now prove that a step of GSIA can only reduce this set, which is then used to prove Proposition 18.

▶ **Definition 19.** *For an SSG G and two strategies (σ, τ), an absorbing set Z is a subset of $V \setminus V_S$ such that starting from any vertex of Z and playing according to (σ, τ), there is a probability zero of reaching a vertex of $V \setminus Z$.*

For σ and τ two strategies, $Z(\sigma, \tau)$ is the set of all vertices in some absorbing set under (σ, τ). Hence, $Z(\sigma, \tau)$ is also an absorbing set. By definition, a play remains stuck in an absorbing set and can never reach a sink, hence all vertices of an absorbing set have *value zero* under (σ, τ). The next lemma proves the existence of the inclusion-wise maximum over τ of $Z(\sigma, \tau)$ that we denote by $Z(\sigma)$. An example is given Fig. 2.

▶ **Lemma 20.** *For every MAX strategy σ, there is $\tau \in BR(\sigma)$ such that for every MIN strategy τ', we have $Z(\sigma, \tau') \subseteq Z(\sigma, \tau)$.*

Proof. For τ in $BR(\sigma)$ and τ' such that $Z(\sigma, \tau') \nsubseteq Z(\sigma, \tau)$, then we define $\tilde{\tau}$ as $\tilde{\tau}(x) = \tau'(x)$ for x in $Z(\sigma, \tau')$ and $\tilde{\tau}(x) = \tau(x)$ otherwise. We now prove that $\tilde{\tau} \in BR(\sigma)$ and $Z(\sigma, \tilde{\tau}) \supseteq Z(\sigma, \tau) \cup Z(\sigma, \tau')$.

Since τ is a best response to σ, we have $v_{\sigma,\tau}(x) \leq v_{\sigma,\tau'}(x)$. Moreover, for $x \in Z(\sigma, \tau')$, $v_{\sigma,\tau'}(x) = 0$ thus $v_{\sigma,\tau}(x) = 0$. From this, we deduce that the two systems of linear equations given by Lemma 8, characterising respectively vectors $v_{\sigma,\tau}$ and $v_{\sigma,\tilde{\tau}}$, are exactly the same: for the only vertices where $\tilde{\tau}(x)$ and $\tau(x)$ differ satisfy $v_{\sigma,\tau}(\tau(x)) = v_{\sigma,\tau}(\tilde{\tau}(x)) = 0$. Hence, we have $v_{\sigma,\tau} = v_{\sigma,\tilde{\tau}}$ and $\tilde{\tau} \in BR(\sigma)$.

For any play under strategies $(\sigma, \tilde{\tau})$ starting in $x \in Z(\sigma, \tau')$, the MIN vertices of the play are all in $Z(\sigma, \tau')$ because $\tilde{\tau}$ plays as τ' on these vertices. Thus, we have $Z(\sigma, \tau') \subseteq Z(\sigma, \tilde{\tau})$. For a play starting in $x \in Z(\sigma, \tau)$, either the play reaches a vertex of $Z(\sigma, \tau')$ and then stays in $Z(\sigma, \tau')$ or it plays like τ and stays in $Z(\sigma, \tau)$. Hence, we have $Z(\sigma, \tau) \subseteq Z(\sigma, \tilde{\tau})$. ◀
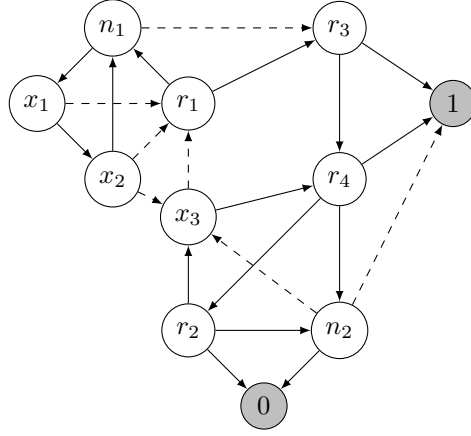
■ **Figure 2** Example of an SSG where the x, n and r vertices are respectively from V_{MAX}, V_{MIN} and V_R. The pair of strategy (σ, τ) is displayed as plain arrows. Here $Z(\sigma, \tau) = Z(\sigma) = \{n_1, x_1, x_2\}$.

From this we deduce the following result on the improvement step for GSIA (where absorbing sets are understood in G):

▶ **Proposition 21** (Proof in Appendix A). *Let G be an SSG, A a set of arcs of G, σ and σ' two* MAX *strategies such that* $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$, *then $Z(\sigma') \subseteq Z(\sigma)$.*

We now prove Proposition 18.

Proof. We introduce a sequence of non-positional strategies $(\sigma_i)_{i \geq 0}$ defined by $\sigma_i = \sigma'|^i \sigma$ for $i \geq 1$. By hypothesis $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$, and by Lemma 16 $v^G_{\sigma'|_A\sigma} = v^{G[A,\sigma]}_{\sigma'}$, then we have

$$v^G_{\sigma_1} = v^G_{\sigma'|\sigma} = v^{G[A,\sigma]}_{\sigma'} > v^{G[A,\sigma]}_\sigma = v^G_\sigma.$$

Hence, by definition, sinks of $G[A, \sigma_1]$ will have at least the values of the corresponding sinks in $G[A, \sigma]$. Applying Lemma 17, we obtain that $v^{G[A,\sigma_1]}_{\sigma'} \geq v^{G[A,\sigma]}_{\sigma'}$, which can also be written as $v^G_{\sigma_2} \geq v^G_{\sigma_1}$. More generally, we have:

$$\forall i \geq 1, v^G_{\sigma_{i+1}} \geq v^G_{\sigma_i} \geq v^G_{\sigma_1} > v^G_\sigma.$$

We now prove that $v^G_{\sigma'} \geq v^G_{\sigma_1}$ to conclude the proof.

From now on, we only consider the game G. Fix a vertex x and a MIN strategy $\tau \in BR(\sigma')$ such that $Z(\sigma') = Z(\sigma', \tau)$. From Proposition 21 we know that, $Z(\sigma') \subseteq Z(\sigma)$. It implies that for every $z \in Z(\sigma')$, $v^G_\sigma(z) = v^G_{\sigma'}(z) = 0$ which implies that $v^{G[A,\sigma]}_{\sigma'}(z) = 0$. Thus, $\sigma'(z) = \sigma(z)$. It implies that $Z(\sigma') \subseteq Z(\sigma, \tau)$.

We now only consider G' the game G where we replace every vertex in $Z(\sigma', \tau)$ by a sink of value 0. Lemma 12 directly implies that $v^G_\sigma = v^{G'}_\sigma$ and $v^G_{\sigma'} = v^{G'}_{\sigma'}$. Moreover, when playing following σ_i when a vertex of $Z(\sigma')$ is reached, for all possible history, the play will stay in the absorbing set. Thus, $v^G_{\sigma_i} = v^{G'}_{\sigma_i}$.

Recall that $\mathbb{P}^x_{\sigma', \tau}(\to s)$ is the probability to reach a sink s in G' while starting in x and following (σ', τ). Let $T^{\sigma', \tau}$ be a random variable defined as the time at which a sink is reached. Note that $T^{\sigma', \tau}$ may be equal to $+\infty$.

For every $i \geq 1$, we use Bayes rule to express the value of $v_{\sigma',\tau}(x)$ while conditioning on finishing the game before i steps.

$$\begin{aligned} v_{\sigma',\tau}(x) =&\mathbb{P}(T^{\sigma',\tau} < i) \sum_{s \in V_S} \mathbb{P}^x_{\sigma',\tau}(\to s \mid T^{\sigma',\tau} < i)\mathrm{Val}(s) \\ &+ \mathbb{P}(i \leq T^{\sigma',\tau} < +\infty) \sum_{s \in V_S} \mathbb{P}^x_{\sigma',\tau}(\to s \mid +\infty > T^{\sigma',\tau} \geq i)\mathrm{Val}(s) \end{aligned}$$

If $T^{\sigma_i,\tau} < i$, only i arcs have been crossed, thus at most i arcs from A have been crossed when the sink is reached. Hence σ_i acts like σ' during the whole play, which yields:

$$\begin{aligned} v_{\sigma',\tau}(x) =&\mathbb{P}(T^{\sigma_i,\tau} < i) \sum_{s \in V_S} \mathbb{P}^x_{\sigma_i,\tau}(\to s \mid T^{\sigma_i,\tau} < i)\mathrm{Val}(s) \\ &+ \mathbb{P}(i \leq T^{\sigma',\tau} < +\infty) \sum_{s \in V_S} \mathbb{P}^x_{\sigma',\tau}(\to s \mid +\infty > T^{\sigma',\tau} \geq i)\mathrm{Val}(s) \end{aligned}$$

We use Bayes rule in the same way for $v_{\sigma_i,\tau}(x)$

$$\begin{aligned} v_{\sigma_i,\tau}(x) =&\mathbb{P}(T^{\sigma_i,\tau} < i) \sum_{s \in V_S} \mathbb{P}^x_{\sigma_i,\tau}(\to s \mid T^{\sigma_i,\tau} < i)\mathrm{Val}(s) \\ &+ \mathbb{P}(i \leq T^{\sigma_i,\tau} < +\infty) \sum_{s \in V_S} \mathbb{P}^x_{\sigma_i,\tau}(\to s \mid T^{\sigma_i,\tau} \geq i)\mathrm{Val}(s) \end{aligned}$$

Since every absorbing vertex in G associated with σ' has been turned into a sink, in G' $\mathbb{P}(T^{\sigma',\tau} < i) = \mathbb{P}(T^{\sigma_i,\tau} < i)$ converges to 1 when i grows. Hence, both $\mathbb{P}(i \leq T^{\sigma',\tau} < +\infty)$ and $\mathbb{P}(i \leq T^{\sigma_i,\tau} < +\infty)$ go to 0 and

$$\lim_{i \to +\infty} |v_{\sigma',\tau}(x) - v_{\sigma_i,\tau}(x)| = 0.$$

Hence, if there was x such that $v_{\sigma'}(x) < v_{\sigma_1}(x)$, we denote $\epsilon = v_{\sigma_1}(x) - v_{\sigma'}(x)$. For some rank I for all $i \geq I$ we have $|v_{\sigma',\tau} - v_{\sigma_i,\tau}| < \epsilon/2$. Which implies $v_{\sigma_i,\tau}(x) < v_{\sigma_1}(x)$. We recall that $v_{\sigma_1}(x) \leq v_{\sigma_i}(x)$. This means that $v_{\sigma_i,\tau} < v_{\sigma_i}(x)$, which contradicts the notion of optimal response against σ_i. Therefore, we have shown that $\sigma' \underset{G}{\geq} \sigma_1 \underset{G}{>} \sigma$. ◄

As a consequence of all previous lemmas, we obtain the correction of GSIA.

▶ **Theorem 22.** *GSIA terminates and returns a pair of optimal strategies.*

Proof. We denote by σ_i the MAX strategy σ at the end of the i-th loop in Algorithm 1. By induction, we prove that the sequence σ_i is of increasing value. Indeed, Line 5 of Algorithm 1 guarantees that $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$, thus Prop. 18 implies that $\sigma' \underset{G}{>} \sigma$, that is $\sigma_{i+1} > \sigma_i$.

The strategies produced by the algorithm are positional, hence there is only a finite number of them. Since the sequence is strictly increasing, it stops at some point. The algorithm only stops when Line 5 of Algorithm 1 fails to find $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$. In other words, σ is optimal in $G[A,\sigma]$. By Lemma 14, σ is also optimal in G. ◄

4   Complexity of GSIA

We analyse the algorithmic complexity of GSIA, by lower bounding the values of the sequence of strategies it produces. We obtain a bound on the number of iterations of GSIA depending on the number of random vertices, rather than on the number of MAX or MIN vertices. Then, we can derive the complexity of any instance of GSIA, by evaluating the cost of computing σ' from σ in $G[A, \sigma]$.

4.1   Values of q-SSGs

To prove a complexity bound using the values of a strategy, we need to precisely characterise the form of these values. In a 2-SSG, there is a function $f(r)$ such that, for every pair of positional strategies (σ, τ), there is $t \leq f(r)$, such that for every vertex x, there is an integer p_x, such that $v_{\sigma, \tau}(x) = \dfrac{p_x}{t}$

Condon proved in [13] that $f(r) \leq 4^r$. Then Auger, Coucheney and Strozecki improved this to $f(r) \leq 6^{r/2}$ in [4]. We show that $f(r) = q^r$ for q-SSGs, which gives the improved bound of $f(r) \leq 2^r$ for 2-SSGs.

▶ **Theorem 23** (Proof in Appendix B). *Let $q \geq 1$ and G a q-SSG with r random vertices, then for any pair of strategies (σ, τ) there is $t \leq q^r$ such that, for every vertex x, there is an integer s_x such that, $v_{\sigma, \tau} = \dfrac{s_x}{t}$.*

Proof of Th. 23 relies on the matrix tree theorem applied to a directed multigraph representing the game under a pair of strategies. Let us show that q^r is a tight bound for $f(r)$. Consider a Markov chain (an SSG with no MAX nor MIN vertices) with $r + 2$ vertices: two sinks 0 and 1 and r random vertices $x_1, \ldots, x_r$. Vertex x_1 goes to 1 with probability $1/q$ and to 0 with probability $(q-1)/q$. For $r \geq i \geq 2$, x_i goes to 0 with probability $(q-1)/q$ and to x_{r-1} with probability $1/q$. Then, the value of x_r is q^{-r}.

4.2   Bounding the Number of Iterations of GSIA

GSIA produces a sequence of strictly increasing positional MAX strategies. The number of positional MAX strategies is bounded by $|\Sigma^{\text{MAX}}| = \displaystyle\prod_{x \in V_{\text{MAX}}} \deg(x)$, hence the number of iterations of GSIA is bounded by this value. If we consider the case of a binary SSG (all vertices of outdegree 2), we have the classical bound of $|\Sigma^{\text{MAX}}| = 2^n$ iterations. The best known bound for a deterministic algorithm is $2^n/n$ iterations obtained for Hoffman-Karp algorithm [25], which is not far from the trivial bound of 2^n iterations.

We give a bound for q-SSG, which depends on q and r the number of random vertices. The difference of two values written as a/b and c/d, with a and b less than q^{-r} is more than q^{-2r}. Hence, if a value increases in GSIA, it increases at least by q^{-2r}. Using the classical notion of switch and anti-switch [25], recalled in Appendix C, we can prove that all vertices which have their value increased by a step of GSIA, are increased by at least q^{-r}.

▶ **Theorem 24** (Proof in Appendix C). *For G a q-SSG with r random vertices and n MAX vertices, the number of iterations of GSIA is at most nq^r.*

The complexity of GSIA is the number of iterations given by Th. 24, multiplied by the complexity of an iteration. In an iteration, there are two sources of complexity: constructing the game $G[A, \sigma]$ and finding an improving strategy σ' in $G[A, \sigma]$. To construct the game, v_σ is computed by solving a linear program of size m up to precision $p = q^r$. Let $C_1(m, p)$ be the

complexity of computing v_σ, then the best bound is currently in $O(m^\omega \log(p))$ [19], with ω the current best bound on the matrix multiplication exponent. Let $C_2(n, r, q)$ be the complexity of computing σ', the complexity of an iteration is in $O(nq^r(C_1(n + r, q^r) + C_2(n, r, q))$.

We obtain a better complexity, when $C_2(n, r, q) = O(C_1(n, q^r)r/n)$, which is the case for most instances of GSIA mentionned in this article. The number of iterations is only rq^r if we can guarantee that a random vertex increases its value at each step. When no random vertex is improved, the cost of computing $G[A, \sigma]$ can be made smaller, which yields the following theorem.

▶ **Theorem 25** (Proof in Appendix D). *Let G be a q-SSG with r random vertices and n MAX vertices. If $C_2(n, r, q) = O(C_1(n, q^r)r/n)$, then the complexity of GSIA is in $O(rq^r C_1(n, q^r))$.*

5 Two Instances of GSIA

As previously mentioned, all known strategy improvement algorithms can be viewed as particular instances of GSIA. This includes e.g. switch-based algorithms, like Hoffman-Karp algorithm [14, 25] or Ludwig's recursive algorithm [22]. With the help of GSIA it also becomes very easy to derive new algorithms, by transforming the game into polynomial time solvable instances, such as almost acyclic games [4]. We detail all these old and new algorithms in Sec. 6.

In this section, we focus instead on two particular instances (or family of instances) of GSIA, for which we obtain new complexity bounds using the results of the previous sections.

5.1 GSIA and f-strategies

The strategy improvement algorithm proposed by Gimbert and Horn in [16] (denoted by GHA) can be viewed as an instance of GSIA where the set A of fixed arcs is the set R of all arcs going out of random vertices, and the improvement step in the subgame $G[R, \sigma]$ consists in taking an optimal strategy. In this case, the subgame $G[R, \sigma]$ is deterministic (random vertices are connected to sinks only and can be replaced by sinks), hence optimal values in $G[R, \sigma]$ depend only on the relative ordering of the values $v_\sigma(x)$ for sink and random vertices x of G. These values can be computed in $O(r \log(r) + n)$ time [1]. In the original paper [16], the algorithm is proposed in a context where the number of sinks is two, but we generalise their definitions to our context.

Consider a total ordering f on $V_R \cup V_S$, $f : x_1 < x_2 < \cdots < x_{r+s}$, where s is the number of sinks. An f-strategy corresponding to this ordering is an optimal MAX strategy in the game where the $s + r$ vertices above are replaced by sinks with new values satisfying $\mathrm{Val}(x_1) < \mathrm{Val}(x_2) < \cdots < \mathrm{Val}(x_{r+s})$. Clearly, this strategy does not depend on the actual values given but only on f. Note that if several f-strategies exist for a given f, they share the same values on all vertices.

Algorithm GHA produces an improving sequence of f-strategies, and the two sinks of value zero and one are always first and last in the order, hence its number of iterations is bounded by $r!$, the total number of possible orderings of the random vertices. We extend this result to a large class of instances of GSIA: let us call Optimal-GSIA (Opt-GSIA), the meta algorithm obtained from Algorithm 1 with two additional constraints:

- the set A of fixed arcs is the same at each step of Algorithm 1;
- at line 5, the improving strategy σ' is the optimal strategy in $G[A, \sigma]$.

All classical algorithms captured by GSIA, or new ones presented in this article are in fact instances of Opt-GSIA. We now show that Opt-GSIA has an iteration number similar to GHA. Since we have proved a bound of nq^r iterations, by Th. 24, Opt-GSIA has essentially the best known number of iterations, for q small and large (the latter being interesting in the case of random vertices with large degree and arbitrary probability distributions).

▶ **Theorem 26.** *Consider an SSG G and a set of arcs A containing k arcs out of* MAX *or* MIN *vertices. Then Algorithm Opt-GSIA runs in at most* $\min((r + k)q^r, (r + k)!)$ *iterations.*

Proof. Let σ be one of the iterated MAX strategies obtained by an instance of Opt-GSIA, and σ' be an optimal strategy in $G[A, \sigma]$. Then σ' is consequently an f-strategy in $G[A, \sigma]$, where f is the ordering on $V_R \cup V_A$ (where V_A is the set of A-sinks) which is induced by the value vector $v_{\sigma'}^{G[A,\sigma]}$ (if vertices have the same value, just arbitrarily decide their relative ordering in f).

Since strategies produced by the algorithm strictly increase in values by Prop. 18, they must be all distinct. Hence, the order f must be distinct at each step of the algorithm, which proves that Opt-GSIA does at most $(r + k)!$ iterations.

Moreover, at every step the value in G of at least one vertex in $V_R \cup V_A$ must improve, by at least q^{-r} because of Th. 23. Since the value of these vertices is bounded by 1, the number of iterations of Opt-GSIA is bounded by $(r + k)q^r$. ◀

Those results can be used to generalize Gimbert and Horn's Algorithm [16] and to compare it with Ibsen-Jensen and Miltersen's algorithm [18]. It is possible to create an instance of GSIA with less iterations than Ibsen-Jensen and Miltersen's algorithm and which terminates exponentially faster on some input. Due to a lack of space this is detailed in a long version of this paper [3].

5.2   Condon's Converge From Below Algorithm

In [14], Condon first presents a faulty algorithm (the Naive Converge From Below Algorithm) and then a correct modified version, the Converge From Below (CFB) Algorithm. This algorithm proceeds by improving a value vector iteratively, but we show here that is in fact a disguised strategy improvement algorithm, that can be seen as an instance of Opt-GSIA. This gives us a proof of convergence of the CFB algorithm in the general, non-stopping case (whereas Condon has the assumption that the game is stopping in her proof), and also bounds on the number of iterations (none are given in the original paper) by Theorem 24 and Theorem 26.

The CFB algorithm is restated with some clarifications on listing 2 (we omit the details of the linear program, see [14]). The algorithm uses two properties of a vector, that we now define. First, vector v is *feasible* if

(i) For $s \in V_S$, $v(s) = \text{Val}(s)$

(ii) For $r \in V_R$, $v(r) = \sum_{x \in N_r} p_r(x)v(r)$

(iii) For $x \in V_{\text{MIN}}$, $v(x) \leq \min_{y \in N^+(x)} v(y)$

(iv) For $x \in V_{\text{MAX}}$, $v(x) \geq \max_{y \in N^+(x)} v(y)$.

A feasible vector is *stable* at x a MIN vertex (resp. MAX vertex) if satisfies condition (iii) (resp. condition (iv)) of feasibility for x with an equality.

We now show by induction that the CFB algorithm is equivalent to the instance of Opt-GSIA where all MIN vertices are fixed, i.e. A is the set of arcs entering MIN vertices. Let A_{MIN} denote this set.

To see this, suppose that at the beginning of line 5 of CFB, Vector v_r is the value vector of a MAX-strategy σ in G. Then:

- at Line 5, we "update v as the feasible vector where all MIN vertices x have value $v_r(x)$ and all MAX vertices are stable". This amounts to finding a MAX-strategy σ' which satisfies optimality conditions in $G[A_{\mathrm{MIN}}, \sigma]$, i.e. an optimal strategy for MAX in this subgame. This is exactly the subgame improvement step of Opt-GSIA. At the end of this step, v is the optimal value vector in $G[A_{\mathrm{MIN}}, \sigma]$;
- in the next loop, at Line 4 of CFB, we "compute the value vector v_r of an optimal response to the MAX strategy that plays greedily according to v", i.e. v_r is updated to the value vector $v_{\sigma'}^G$. This is precisely Line 6 of GSIA when we update values in the subgame.

Hence, we see that except for the initialisation where v_r may not correspond to a MAX-strategy, it will be the case as soon as we reach Line 5 of the first loop, and from this point on CFB will correspond exactly to the instance of Opt-GSIA described above.

Algorithm 2 Converge From Below Algorithm.

Data: G an SSG
Result: The optimal value vector v^* of G
1 begin
2 · let v be a feasible vector in which all MIN vertices have value 0 and all MAX vertices are stable
3 **while** v *is not an optimal value vector* **do**
4 · use linear programming to compute the value vector v_r of an optimal response to the MAX strategy that plays greedily according to v
5 · update v as the feasible vector where all MIN vertices x have value $v_r(x)$ and all MAX vertices are stable
6 **return** v

---

6 Algorithms Derived from GSIA

We show that all known strategy improvement algorithms can be expressed as instances of GSIA and we also propose several new algorithms, derived from choices of A which make the transformed game polynomial time solvable. The only algorithms which are not instances of GSIA are based on values rather than strategies: value propagation [10, 14, 18], quadratic programming [14, 21] and dichotomy [4].

6.1 Hoffman-Karp Algorithms

The most classical method to solve an SSG, called the Hoffman-Karp algorithm, repeatedly applies switches to the strategy until finding the optimal one. It is also a generic algorithm, since the choice of the set of vertices to switch at each step is not specified nor the choice of the initial strategy. Many details on these algorithms can be found in [14] or [25].

Hoffman-Karp algorithms are instances of GSIA, where A is the set of all arcs of the SSG. Indeed, as proved in Lemma 30, a switch σ' of σ satisfies $\sigma \underset{G[A,\sigma]}{\succ} \sigma'$. Interpreting Hoffman-Karp algorithms as instances of GSIA proves that they work on non-stopping games, while in most article the stopping condition is required. Moreover, it shows that their number of iterations is $O(nq^r)$ on q-SSGs, a complexity exponential in r only, which was known only for algorithms specially designed for this purpose [17, 15, 18, 5].

Ludwig's Algorithm [22], which is the best randomised algorithm to solve SSGs, can be seen as an Hoffman-Karp algorithm using Bland's rule as shown in [5]: a random order on the vertices is drawn, and at each step, the first switchable vertex in the order is switched. Two other Hoffman-Karp algorithms are presented in [25]: switching all switchable vertices at each step or switching a random subset. Seeing these three algorithms as instances of GSIA yields $O(nq^r)$ as a deterministic bound on their number of iterations, which was unknown. However, the analysis of [22, 25] is required to obtain a good complexity in n for these algorithms.

6.2 Selection of the Initial Strategy

In [15], Dai and Ge give a randomised improvement of GHA simply by choosing a better initial strategy. To do so, they choose randomly $\sqrt{r!}\log(r!)$ strategies and choose the one with the highest value. This ensures, with high probability, that at most $\sqrt{r!}$ iterations will be done in GHA. Thus, their algorithm runs in $O\left(\left(\sqrt{r!}\right)\right)$ iterations. This algorithm is also captured by GSIA by selecting the initial strategy in the same way, however it seems hard to combine the gain made by the random selection of the strategy and the bound in $O(q^r)$ of GSIA, since even a strategy close to the optimal one may have values far from it. Remark that it is trivial to extend this method to any instance of Opt-GSIA to improve on the complexity of Th. 26.

6.3 New Algorithms

We can use GSIA to design many strategy improvement algorithms. We present three of them, all based on a choice of A which makes $G[A, \sigma]$ solvable in polynomial time. The initial strategy can be anything and σ' is always chosen to be the optimal strategy in $G[A, \sigma]$. Most of them can be seen as generalisations of known algorithms.

1. Let A be a feedback arc set of G, then $G[A, \sigma]$ is acyclic and it can be solved in linear time. It seems intuitively appealing to think that this algorithm will be faster if the feedback arc set is small but we have no proof to sustain such a proposition.
2. A MAX acyclic SSG is an SSG such that that every MAX vertex has at most one outgoing arc in a cycle. MAX acyclic SSG can be solved in polynomial time, see [4]. If we let A be a set of arc that contains all but one outgoing arcs of each MAX vertex, then $G[A, \sigma]$ is MAX acyclic and can be solved in polynomial time. Moreover, such a game can be solved by strategy improvement in at most n iterations. This can be seen as a generalisation of Hoffman-Karp algorithm, in which A contains *all* outgoing arcs of MAX vertices.
3. As an intermediate between acyclic games and MAX acyclic games, we may consider almost acyclic games, where all vertices have at most one outgoing arc in a cycle. Almost acyclic SSGs can be solved in linear time [4].

—— **References** ——————————————————————————

1   Daniel Andersson, Kristoffer Arnsfelt Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. Deterministic graphical games revisited. In *Conference on Computability in Europe*, pages 1–10. Springer, 2008.
2   Daniel Andersson and Peter Bro Miltersen. The complexity of solving stochastic games on graphs. In *International Symposium on Algorithms and Computation*, pages 112–121, 2009.
3   David Auger, Xavier Badin de Montjoye, and Yann Strozecki. A generic strategy iteration method for simple stochastic games. *arXiv preprint*, 2021. `arXiv:2102.04922`.

4 David Auger, Pierre Coucheney, and Yann Strozecki. Finding optimal strategies of almost acyclic simple stochastic games. In *International Conference on Theory and Applications of Models of Computation*, pages 67–85, 2014.

5 David Auger, Pierre Coucheney, and Yann Strozecki. Solving Simple Stochastic Games with Few Random Nodes Faster Using Bland's Rule. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, pages 9:1–9:16, 2019.

6 Cristian S Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasi-polynomial time. *SIAM Journal on Computing*, 0(0):STOC17–152, 2020.

7 Seth Chaiken and Daniel J Kleitman. Matrix tree theorems. *Journal of combinatorial theory, Series A*, 24(3):377–381, 1978.

8 Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Strategy improvement for concurrent reachability and turn-based stochastic safety games. *Journal of Computer and System Sciences*, 79(5):640–657, 2013.

9 Krishnendu Chatterjee and Nathanaël Fijalkow. A reduction from parity games to simple stochastic games. *Electronic Proceedings in Theoretical Computer Science*, 54, 2011.

10 Krishnendu Chatterjee and Thomas A. Henzinger. *Value Iteration*, pages 107–138. Springer Berlin Heidelberg, 2008.

11 Taolue Chen, Marta Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. Synthesis for multi-objective stochastic games: An application to autonomous urban driving. In *Quantitative Evaluation of Systems*, pages 322–337, 2013.

12 Thomas Colcombet and Nathanaël Fijalkow. Universal graphs and good for games automata: New tools for infinite duration games. In *International Conference on Foundations of Software Science and Computation Structures*, pages 1–26. Springer, 2019.

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

14 Anne Condon. On algorithms for simple stochastic games. *Advances in computational complexity theory*, 13:51–73, 1993.

15 Decheng Dai and Rong Ge. New results on simple stochastic games. In *International Symposium on Algorithms and Computation*, pages 1014–1023. Springer, 2009.

16 Hugo Gimbert and Florian Horn. Simple stochastic games with few random vertices are easy to solve. In *Foundations of Software Science and Computational Structures*, pages 5–19. Springer, 2008.

17 Hugo Gimbert and Florian Horn. Solving simple stochastic games with few random vertices. *Logical Methods in Computer Science*, Volume 5, Issue 2, 2009.

18 Rasmus Ibsen-Jensen and Peter Bro Miltersen. Solving simple stochastic games with few coin toss positions. In *European Symposium on Algorithms*, pages 636–647. Springer, 2012.

19 Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. *arXiv preprint*, 2020. `arXiv:2004.07470`.

20 Brendan Juba. On the hardness of simple stochastic games. *Master's thesis, CMU*, 2005.

21 Jan Křetínský, Emanuel Ramneantu, Alexander Slivinskiy, and Maximilian Weininger. Comparison of algorithms for simple stochastic games. *arXiv preprint*, 2020. `arXiv:2009.10882`.

22 Walter Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and computation*, 117(1):151–155, 1995.

23 L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.

24 C Stirling. Bisimulation, modal logic and model checking games. *Logic Journal of the IGPL*, 7(1):103–124, 1999.

25 Rahul Tripathi, Elena Valkanova, and VS Anil Kumar. On strategy improvement algorithms for simple stochastic games. *Journal of Discrete Algorithms*, 9(3):263–278, 2011.

A   Proof of the decreasing absorbing set

Here we prove Prop. 21

Proof. Suppose that $Z(\sigma')$ is not a subset of $Z(\sigma)$. From Lemma 20 there is $\tau \in BR(\sigma)$ such that $Z(\sigma, \tau) = Z(\sigma)$ and $\tau' \in BR(\sigma')$ such that $Z(\sigma', \tau') = Z(\sigma')$. We write $Z = Z(\sigma')$

Let X be the set of MAX vertices x in $Z(\sigma') \setminus Z(\sigma)$ such that $\sigma(x) \neq \sigma'(x)$; it is nonempty otherwise $Z(\sigma')$ would be an absorbing set for (σ, τ'). If x is in X, since $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$, we have

$$v_{\sigma'}^{G[A,\sigma]}(x) > v_{\sigma}^{G[A,\sigma]}(x) \geq 0$$

Thus, a sink is reached in $G[A, \sigma]$ starting from x under the strategies (σ', τ'). Since Z is an absorbing set in G under the same strategies, it implies that all the accessible sinks in $G[A, \sigma]$ are A-sinks. Hence, there is at least one arc $e = (y, z) \in A$ with both ends in Z and such that $v_\sigma(z) > 0$. We define the vertex s of Z as:

$$s = \arg\max_{z \in Z}\{v_\sigma(z) \mid \exists y \in V, (y, z) \in A\}$$

and we let $v = v_\sigma(s)$. The value of each vertex in Z is bounded by v. Similarly than for x, in G under strategies (σ, τ) the value of s is bounded by the value of the vertex leaving Z. Such vertices exist since Z is not a subset of $Z(\sigma)$. We now want to show that those vertices all have value strictly lesser than v, thus proving a contradiction.

First, since Z is an absorbing set for (σ', τ'), all arcs leaving a random vertex in $Z(\sigma')$ remain in $Z(\sigma')$ in G; this is not dependent on the strategies considered.

Let $E_X \subseteq X$ the set of MAX vertices x of X such that $\sigma(x) \notin Z$ and let $E_N \subseteq Z \cap V_{\text{MIN}}$ the set of MIN vertices x of Z such that $\tau(x) \notin Z$.

On the one hand, for a MIN vertex $x \in E_N$:

$$
\begin{aligned}
v_\sigma^G(\tau(x)) &\leq v_\sigma^G(\tau'(x)) && \text{Since } \tau = \tau(\sigma) \\
v_\sigma^G(\tau'(x)) &= v_\sigma^{G[A,\sigma]}(\tau'(x)) && \\
v_\sigma^{G[A,\sigma]}(\tau'(x)) &\leq v_{\sigma'}^{G[A,\sigma]}(\tau'(x)) && \text{Since } \sigma' \underset{G[A,\sigma]}{\succ} \sigma \\
v_{\sigma'}^{G[A,\sigma]}(\tau'(x)) &\leq v && \text{Since } \tau'(x) \in Z
\end{aligned}
$$

Thus, $v_\sigma^G(\tau(x)) \leq v$. In case of equality, we have $v = v_\sigma^G(\tau'(x)) = v_\sigma^G(\tau(x))$; hence we can replace τ by $\bar{\tau}(x)$, which is identical to τ except that $\bar{\tau}(x) = \tau'(x)$. We have $v_{\sigma,\tau} = v_{\sigma,\bar{\tau}}$ and $Z(\sigma, \tau) = Z(\sigma, \bar{\tau})$. Indeed, according to Lemma 8 the only situation that could occur would be to violate the condition (v) by creating an absorbing set. However this would contradict the definition of τ. Thus, we can suppose that for any x in E_N, $v_\sigma(\tau(x)) < v$.

On the other hand, since $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$ we know that for any x in E_X:

$$v_\sigma^G(\sigma(x)) < v_{\sigma'}^{G[A,\sigma]}(x) \leq v$$

Now, for any vertex x of $E = E_X \cup E_N$, let p_x be the probability of x being the first vertex of E reached starting from s following strategies (σ, τ). By conditional expectation:

$$v_\sigma(s) = \sum_{x \in E_X} p_x v_\sigma(\sigma(x)) + \sum_{x \in E_N} p_x v_\sigma(\tau(x))$$

Thus, $v_\sigma(s) < v$ which contradicts the definition of v, and proves that $Z(\sigma') \subseteq Z(\sigma)$. ◄

B Proof of the value of a q-SSG

Here, we prove Th. 23.

Let us remark that a q-SSG can be assumed to have all its probability transition of the form p/q. The idea here is to notice that it is possible to loop with a certain probability on the same random vertices.

▶ **Lemma 27.** *Let G be a q-SSG, then there is G' a q-SSG with the same vertices which defines the same expectation $\mathbb{E}_{\sigma,\tau}^{x_0}(\cdot|\cdot)$ and such that for all $x \in V_R$ and all $x' \in N^+(x)$ then there is an integer $p_{x,x'}$ such that $p_x(x') = p_{x,x'}/q$.*

Proof. For a a random vertex in G, and $q_a < q$ such that for every other vertex x in G there is $p_x \in \mathbb{N}$ and a probability p_x/q_a to go directly from a to x, we change those probabilities to p_x/q and we add a probability p/q to stay in a, where:

$$p = q - \sum_{x \in V} p_x. \qquad \blacktriangleleft$$

Now, we state the classical matrix-tree theorem that we use in our proof (see e.g. [7]). Let G be a directed multigraph with n vertices, then the *Laplacian matrix* of $\mathcal{G}$ is a $n \times n$ matrix $L(\mathcal{G}) = (l_{i,j})_{i,j \leq n}$ defined by:

  (i) $l_{i,j}$ equals $-m$ where m is the number of arcs from i to j.

  (ii) $l_{i,i}$ is the number of arcs going to i, excluding the self-loops.

▶ **Theorem 28** (Matrix tree theorem for directed multigraphs). *For $G = (V, E)$ a directed multigraph with vertices $V = \{v_1, \ldots, v_k\}$ and L its Laplacian matrix, the number of spanning trees rooted at v_i is $\det(\hat{L}_{i,i})$ where $\hat{L}_{i,i}$ is the matrix obtained by deleting the i-th row and column from L.*

We can now prove Th. 23.

Proof of Th. 23. The beginning of the proof is the same as in [14] and [4]. We start by transforming the game with fixed strategies in a Markov Chain with equivalent values. Then, we show that the value of each vertex can be written $\dfrac{\det B_i}{\det q(I - A)}$ using Cramer rule, for B_i and A two matrix which will be carefully defined. To conclude, we will show that $\det q(I - A) < q^r$ by creating a graph obtain from our initial game and using Th. 28.

We consider a q-SSG G and two positional strategies σ and τ. Without loss of generality, we can restrict ourselves to the computation of non-zero, non-sink values. Thus, each vertex has a non-zero probability to reach the 1-sink. To compute the values $v_{\sigma,\tau}$, we can consider G_A an SSG with vertices $V_R \cup V_S$: the random vertices and the sinks of V. The value of the sinks is not changed and the probability distribution p'_x is defined as follows. For $x \in V_R$ and x' in G_A, we call $M_{x,x'}$ the set of MAX and MIN vertex y in $N^+(x)$ such that there is a path following only arcs of σ and τ from y to x'. We then have

$$p'_x(x') = \sum_{y \in M_{x,x'}} p_x(y)$$

The graph G_A has $r + 2$ vertices that we denote by $a_1, \ldots, a_{r+1}, a_{r+2}$ where a_{r+1} is the 0-sink and a_{r+2} is the 1-sink. Let b be the r-dimensional column vector with $b_i = p'_{a_i}(a_{r+2})$. We define A the $r \times r$ matrix, with $A_{i,j} = p'_{a_i}(a_j)$.

The values of the random vertices are defined by the vector z that satisfies the following equation:

$$z = Az + b$$

Let I be the identity matrix, $(I - A)$ is invertible because each random vertex has access to a sink and every eigenvalue of A is strictly less than 1. We refer to [14] for details. Hence, the equation has a unique solution and z is also solution of:

$$q(I - A)z = qb$$

Hence, under the strategies σ, τ, the value z_i of a random vertex a_i given by the Cramer rule is

$$z_i = \frac{\det B_i}{\det q(I - A)}$$

where B_i is the matrix $q(I - A)$ where the i-th column has been replaced by qb. The value $\det B_i$ is an integer. See [4] for more details. Our goal is now to bound $\det q(I - A)$.

From the graph G_A, we construct the graph G' by inverting all arcs, and duplicating an arc of probability p/q into p arcs of probability $1/q$. We also add an arc coming from the 1-sink to the 0-sink and one from the 0-sink towards the 1-sink. Figure 3 shows an example of the transformation from G to G'. The Laplacian L of G' is thus the following matrix.

$$L = \left(\begin{array}{c|cc} q(I - A)^T & \multicolumn{2}{c}{B} \\ \hline \multirow{2}{*}{0} & 1 & 1 \\ & 1 & 1 \end{array} \right)$$

Indeed, every random vertex has indegree q minus the number of loops. Thus the number of spanning trees of G' rooted in the 1-sink is equal by Th. 28 to $\det \hat{L}_{r+2,r+2}$ where we have

$$\hat{L}_{r+2,r+2} = \left(\begin{array}{c|c} q(I - A)^T & B' \\ \hline 0 & 1 \end{array} \right).$$

In other words, the number of spanning trees of G' is equal to $\det q(I - A)$. Furthermore, each spanning tree contains exactly one incoming arcs for every random vertices, and the arc (a_{r+2}, a_{r+1}) has to be used. Thus, there is at most q^r spanning trees rooted in G' and $\det q(I - A) \leq q^r$. ◀

C   Bounding the Number of Iterations of GSIA

We introduce the notion of switch and anti-switch, to prove that the improvement is at least q^{-r} rather than q^{-2r}.

▶ **Definition 29.** *A switch (resp. an anti-switch) of a* MAX *strategy σ with switched set $S \subseteq V_{\text{MAX}}$ is a strategy σ_S defined by $\sigma_S(x) = \sigma(x)$ for $x \notin S$, and satisfying $v_\sigma(\sigma(x)) < v_\sigma(\sigma_S(x))$ (resp. $v_\sigma(\sigma(x)) > v_\sigma(\sigma_S(x))$) for $x \in S$ (hence $\sigma_S(x) \neq \sigma(x)$).*

A common tool to solve SSGs is the fact that a switch increases the value of a strategy, while an anti-switch decreases it. Within our framework of transformed game, it is extremely simple to prove.

▶ **Lemma 30.** *If σ_S is a switch of σ, then $\sigma_S > \sigma$. If σ_S is an anti-switch of σ, then $\sigma_S < \sigma$.*
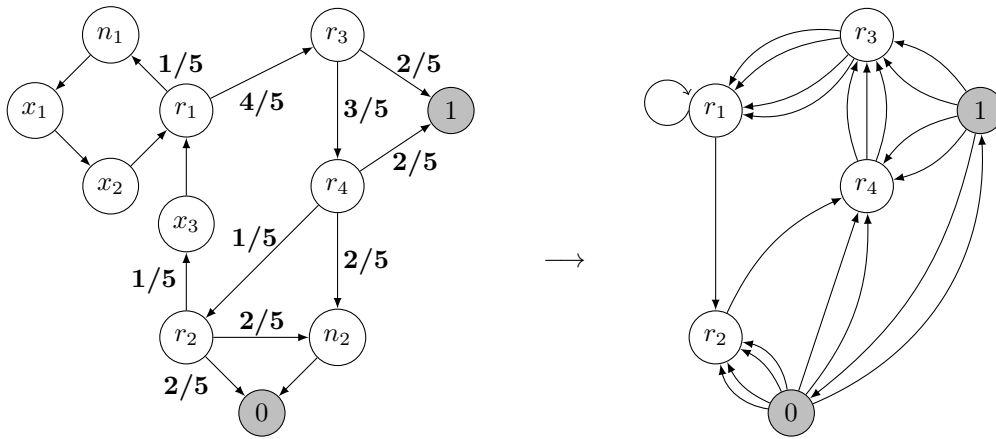
Figure 3 Example of a transformation of a graph G into a graph G'.

Proof. Consider $G[A, \sigma]$ the game obtained from G, where A is the set of all arcs of G. Let us consider x a vertex switched in σ', that is with $v_\sigma^G(\sigma(x)) < v_\sigma^G(\sigma'(x))$. Then, because all arcs are in A, we have $v_\sigma^{G[A,\sigma]}(x) = v_\sigma^G(\sigma(x))$ and $v_{\sigma'}^{G[A,\sigma]}(x) = v_\sigma^G(\sigma'(x))$. Hence, $v_\sigma^{G[A,\sigma]}(x) < v_{\sigma'}^{G[A,\sigma]}(x)$ and for $v_\sigma^G(\sigma(x)) \geq v_\sigma^G(\sigma'(x))$, $\sigma(x) = \sigma'(x)$, which implies $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$. Prop. 18 proves $\sigma' \underset{G}{>} \sigma$.

The proof is the same for an anti-switch, since $\sigma \underset{G[A,\sigma]}{\succ} \sigma' \Rightarrow \sigma \underset{G}{>} \sigma'$ (which can be proved similarly as Prop. 18, while keeping in mind that in the decreasing case, creating absorbing set lowers the value). ◀

We use the previous lemma to prove Th. 24.

Proof. Let us consider σ the strategy computed at some point by GSIA and σ' the next strategy. By Prop. 18, $\sigma < \sigma'$. Hence, by Lemma 30, σ' cannot be an anti-switch of σ. Thus, there is a MAX vertex x such that $v_\sigma(\sigma(x)) < v_\sigma(\sigma'(x))$. We recall that $\sigma'(x)$ denotes the successor of x under strategy σ'.

Since $\sigma < \sigma'$, we have $v_\sigma(x) = v_\sigma(\sigma(x)) < v_\sigma(\sigma'(x)) \leq v_{\sigma'}(\sigma'(x)) = v_{\sigma'}(x)$. We now evaluate $v_\sigma(\sigma'(x)) - v_\sigma(\sigma(x))$. In the game G, under the strategies $\sigma, \tau(\sigma)$, Th. 23 implies that for some $t \leq q^r$, $v_\sigma(\sigma(x)) = p/t$ and $v_\sigma(\sigma'(x)) = p'/t$. We have $p/t < p'/t$, thus $p'/t - p/t \geq 1/t \geq 1/q^r$. Hence, the value of some MAX vertex increases by $1/q^r$ in each iteration of GSIA. Since there are n MAX vertices and their values are bounded by 1, there are at most nq^r iterations. ◀

D   Amortised Complexity of GSIA

Here, we prove Th. 25.

Proof. We assume that $r < n$, otherwise the theorem is trivial. Let σ' be the strategy computed by GSIA at some point, improving on the strategy σ. GSIA must compute $G[A, \sigma']$, and thus $v_{\sigma'}$ and we explain a method to do so efficiently.

We assume that the order of the values (in G) of the random vertices is the same for σ and σ'. Then, knowing this order and σ', it is easy to compute $\tau(\sigma')$ a best response to σ' in $O(r \log(r) + n)$ time [1]. Then, we can compute the values $v_{\sigma', \tau(\sigma')}$ in time $O(C_1(r, q^r))$, since it is done by solving a linear system of dimension r with precision q^r, a task which

is simpler than solving a linear program. Since $C_1(r, q^r)$ is at least quadratic in r, then $C_1(r, q^r) < C_1(n, q^r)r/n$ and by hypothesis $C_2(n, r, q) = O(C_1(n, q^r)r/n)$, hence a step is of complexity at most $O(C_1(n, q^r)r/n)$. There are at most nq^r such steps, for a total complexity of $O(rq^r C_1(n, q^r))$.

We need to detect when the assumption that the values of the random vertices are the same for σ and σ' is false. If $v_{\sigma', \tau(\sigma')}$ satisfies the optimality conditions at the MIN vertices, then $\tau(\sigma')$ is a best response. Otherwise, we compute the best response by solving a linear program in time $C(n, q^r)$. In that case, the order of the random vertices has changed: there are two vertices x_1 and x_2 such that $v_\sigma(x_1) < v_\sigma(x_2)$ and $v_{\sigma'}(x_1) > v_{\sigma'}(x_2)$. Hence, $v_{\sigma'}(x_1) > v_\sigma(x_2)$, which implies that $v_{\sigma'}(x_1) - v_\sigma(x_1) > v_\sigma(x_2) - v_\sigma(x_1) > q^{-r}$.

We have proved that when the random order changes, the value of some random vertex increases by at least q^{-r}, hence there are at most rq^r such steps. The complexity from these steps is bounded by $O(rq^r C_1(n, q^r))$, which proves the theorem.   ◀