Generic Strategy Improvement for Simple Stochastic Games

David Auger, Xavier Badin de Montjoye and Yann Strozecki

Université de Versailles St-Quentin-en-Yvelines
Laboratoire DAVID
Versailles, France

Mars 2022, Séminaire LACL

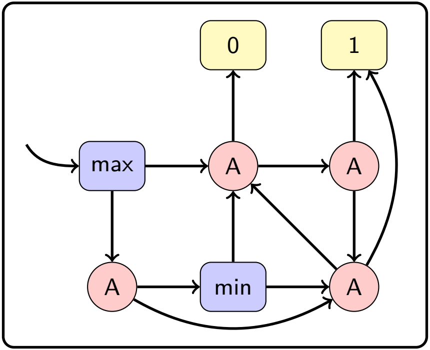
What's an SSG ?

Yet another game

Simple stochastic game (SSG)

A Simple Stochastic Game (Shapley, Condon) is defined by a directed graph with:

- three sets of vertices V_{MAX}, V_{MIN}, V_{AVE} of outdegree 2
- two (or more) 'sink' vertices with values 0 and 1

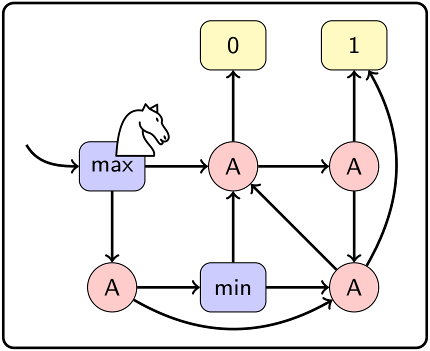


Two players: MAX and MIN, and *randomness*.

Rules of an SSG

A play consists in moving a *pebble* on the graph:

- player MAX wants to maximize the value of the sink reached.
- player MIN wants to minimize the value. If no sink is reached, the value is 0.

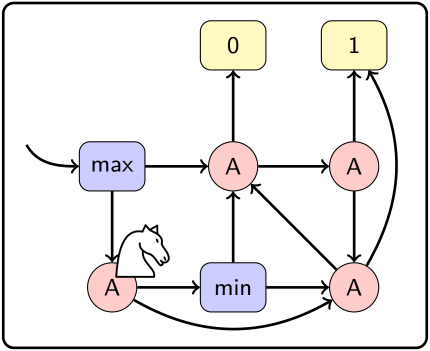


On a MAX node player MAX decides where to go next.

Rules of an SSG

A play consists in moving a *pebble* on the graph:

- player MAX wants to maximize the value of the sink reached.
- player MIN wants to minimize the value. If no sink is reached, the value is 0.

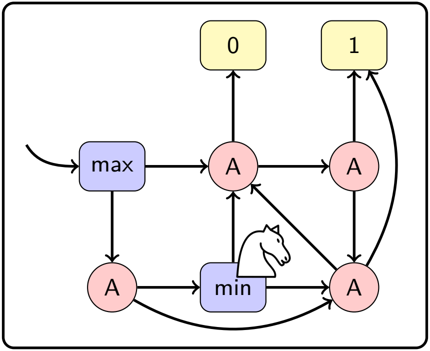


On a AVE node the next vertex is randomly determined.

Rules of an SSG

A play consists in moving a *pebble* on the graph:

- player MAX wants to maximize the value of the sink reached.
- player MIN wants to minimize the value. If no sink is reached, the value is 0.

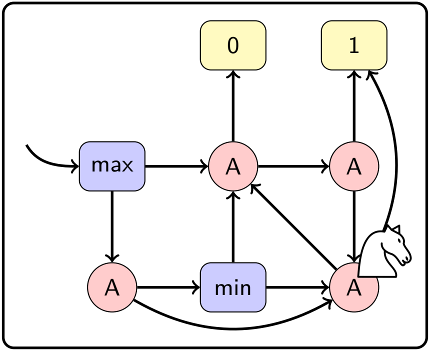


On a MIN node player MIN decides where to go next.

Rules of an SSG

A play consists in moving a *pebble* on the graph:

- player MAX wants to maximize the value of the sink reached.
- player MIN wants to minimize the value. If no sink is reached, the value is 0.

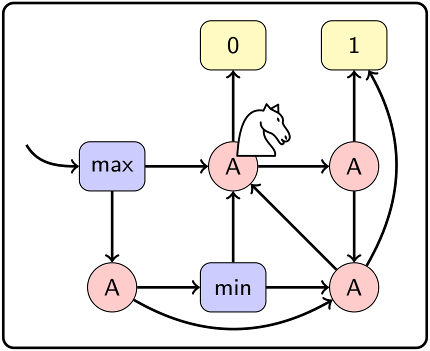


Etc.

Rules of an SSG

A play consists in moving a *pebble* on the graph:

- player MAX wants to maximize the value of the sink reached.
- player MIN wants to minimize the value. If no sink is reached, the value is 0.

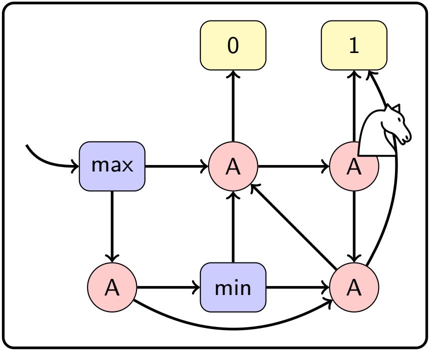


Etc.

Rules of an SSG

A play consists in moving a *pebble* on the graph:

- player MAX wants to maximize the value of the sink reached.
- player MIN wants to minimize the value. If no sink is reached, the value is 0.

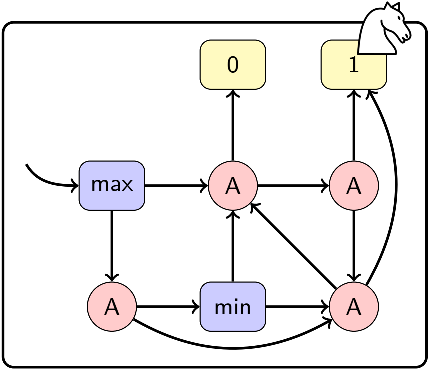


Etc.

Rules of an SSG

A play consists in moving a *pebble* on the graph:

- player MAX wants to maximize the value of the sink reached.
- player MIN wants to minimize the value. If no sink is reached, the value is 0.

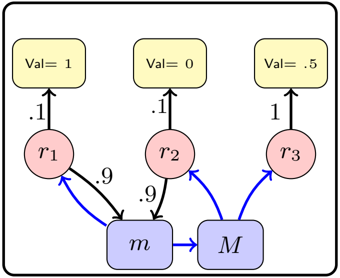


Etc.

Generalized SSGs

Generalize *binary* SSG:

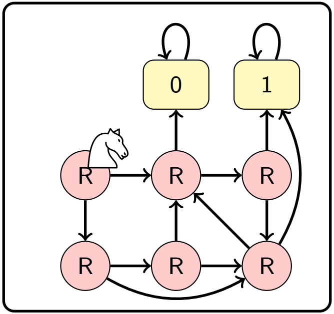
- arbitrary outdegree on the MAX and MIN nodes
- arbitrary values on sinks
- arbitrary probability distribution on the outneighbours of each AVE node



What's the value of an SSG ?

Markov Chain

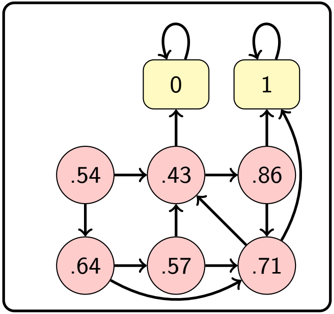
- a finite, stationnary Markov chain as a collection of *random nodes* with a token moving
- stopping condition: proba 1 of reaching a *sink node*, each with a given *value*



***value* of node v = *average value* of the sink that is reached**

Values of nodes

Here : binary case (outdegree 2, uniform probability)



Easily computed by linear system :

$$\forall \text{ non sink node } v, \quad val[v] = \sum_{w} p(v,w) \cdot val[w]$$

Form of the Values

We assume that the probability distribution on each random vertex has values of the form p/q, for some q. For binary Markov chain, $q = 2$.

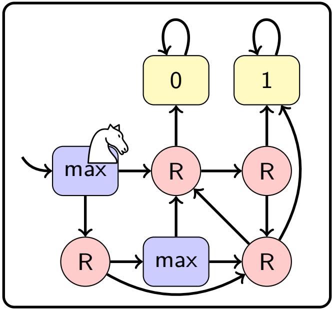
Value format

In a Markov chain with r vertices, there is $t \leq q^r$ such that each vertex v has value $\frac{p_v}{t}$.

This is proven using the **matrix tree theorem**.

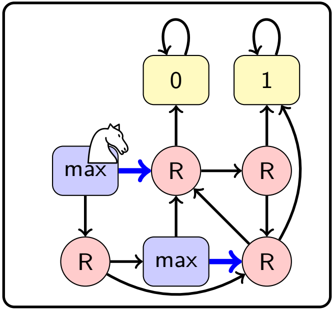
Markov Decision Process

- Add some *decision nodes* and 1 player
- On a decision node, the player chooses the next node among neighbours



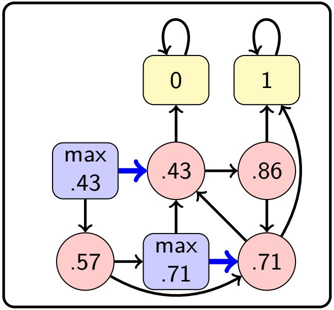
goal : maximize the *value* of a node / all nodes

Markovian property in MDP



- There is an optimal solution which is stationnary and pure (deterministic)
- *strategy* := choice of an outneighbour for every max node

Values of a strategy



- There is an optimal solution which is stationnary and pure (deterministic)
- *strategy* := choice of an outneighbour for every max node
- Values obtained from the underlying Markov chain

Solving a MDP

Bellman equations for optimal values val_* (under mild conditions)

- $\forall v$ random node

$$val_*[v] = \sum_w p(v,w) \cdot val_*[w]$$

- $\forall$ max node

$$val_*[v] = \max_{(v,w)\in A} val_*[w]$$

- max / linear system
- solved by LP in polynomial time

Optimal values in an SSG

We consider only *positional strategies*:

$$\sigma : V_{\mathsf{MAX}} \longrightarrow V, \quad \tau : V_{\mathsf{MIN}} \longrightarrow V$$

The *value* of a vertex x is the best expected value of a sink that MAX can guarantee starting from x:

$$val_*(x) = \max_{\substack{\sigma \text{ strategy} \\ \text{for MAX}}} \; \min_{\substack{\tau \text{ strategy} \\ \text{for MIN}}} \; \underbrace{\mathbb{E}_{\sigma,\tau}\left(\text{value of the sink reached} \;\middle|\; \text{game starts in } x\right)}_{val_{\sigma,\tau}(x)}$$

Problem: given a game and a vertex, compute the value of the vertex.

Decision problem: $val_*(x) > 0.5$?

Alternative version: find the pair of optimal strategies (σ^*, τ^*)

Optimal values in an SSG

We consider only *positional strategies*:

$$\sigma : V_{\mathsf{MAX}} \longrightarrow V, \quad \tau : V_{\mathsf{MIN}} \longrightarrow V$$

The *value* of a vertex x is the best expected value of a sink that MAX can guarantee starting from x:

$$val_*(x) = \max_{\substack{\sigma \text{ strategy} \\ \text{for MAX}}} \min_{\substack{\tau \text{ strategy} \\ \text{for MIN}}} \underbrace{\mathbb{E}_{\sigma,\tau}\left(\text{value of the sink reached} \;\middle|\; \text{game starts in } x\right)}_{val_{\sigma,\tau}(x)}$$

Problem: given a game and a vertex, compute the value of the vertex.

Decision problem: $val_*(x) > 0.5$?

Alternative version: find the pair of optimal strategies (σ^*, τ^*)

Solving an SSG

Bellman equations for optimal values val_* (under mild conditions)

- $\forall v$ random node
$$val_*[v] = \sum_w p(v,w) \cdot val_*[w]$$
- $\forall v$ MAX node
$$val_*[v] = \max_{(v,w)\in A} val_*[w]$$
- $\forall v$ MIN node
$$val_*[v] = \min_{(v,w)\in A} val_*[w]$$

- max / min / linear system
- there is $t \leq q^r$ such that for each vertex v, $val_*[v] = \frac{p_v}{t}$
- Complexity somewhere between P and $\mathrm{EOPL} = \mathrm{PLS} \cap \mathrm{PPAD}$
- Harder than *Parity Game, Mean payoff Game, Discounted payoff Game* but equivalent to their stochastic versions.

Solving an SSG

Bellman equations for optimal values val_* (under mild conditions)

- $\forall v$ random node
$$val_*[v] = \sum_w p(v,w) \cdot val_*[w]$$
- $\forall v$ MAX node
$$val_*[v] = \max_{(v,w)\in A} val_*[w]$$
- $\forall v$ MIN node
$$val_*[v] = \min_{(v,w)\in A} val_*[w]$$

- max / min / linear system
- there is $t \leq q^r$ such that for each vertex v, $val_*[v] = \frac{p_v}{t}$
- Complexity somewhere between P and $\mathrm{EOPL} = \mathrm{PLS} \cap \mathrm{PPAD}$
- Harder than *Parity Game, Mean payoff Game, Discounted payoff Game* but equivalent to their stochastic versions.

Algorithms to solve SSGs

Classical Methods

Several methods to compute the value of an SSG:

- *Quadratic programming*, express min and max constraints as a sum of quadratic functions to minimize
- *Value iteration*, apply a contracting operator on the values
- *Dichotomic search*, find the values by dichotomic search
- *LP-type problem*
- *Unique sink orientation*
- *Strategy improvement*

Strategy improvement algorithm

Strategy improvement algorithm: sequence of MAX-strategies of strictly increasing values.
n: number of MAX-vertices
r: number of random vertices
Binary SSG: MAX, MIN and average vertices of degree two

Algorithms based on switches

- Hoffman-Karp's algorithm $\rightarrow$ binary SSGs, $O\left(2^n/n\right)$ iterations
- Fibonnaci Seesaw algorithm $\rightarrow$ binary SSGs, $O\left(1.61^n\right)$ iterations
- Gimbert-Horn's algorithm $\rightarrow$ $O\left(r!\right)$ iterations
- Ludwig's algorithm $\rightarrow 2^{O(\sqrt{n})}$ expected iterations
- Auger, Coucheney, Strozecki's algorithm $\rightarrow 2^{O(r)}$ expected iterations

Game with shortcuts

Game transformation

A: a subset of arcs of G and σ a MAX strategy.
$G[A, \sigma]$: copy of G with a modification $\longrightarrow$ each arc $e = (x, y) \in A$ removed and replaced by $e' = (x, s_e)$
s_e: new sink vertex with value $v_{\sigma,\tau(\sigma)}(y)$.

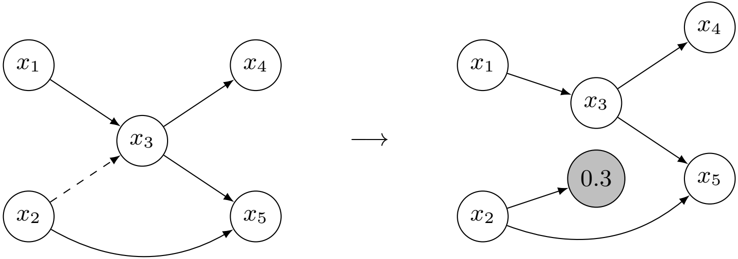


Figure: Transformation of G in $G[\{(x_2, x_3)\}, \sigma]$ with $v_\sigma(x_3) = 0.3$

The generic strategy improvement algorithm (GSIA)

Order of strategies

Let σ, σ' be two MAX-strategies, $\sigma \succ \sigma'$ iff $v_\sigma > v_{\sigma'}$ and for all MAX-vertices x such that $v_\sigma(x) = v_{\sigma'}(x)$, we have $\sigma(x) = \sigma'(x)$.

Algorithm 1: GSIA

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

A generic algoithm

A generic algorithm

Three choices:

- the initial strategy
- the set A of fixed arcs
- how to find σ'

The set of arcs and the method to find σ' can change at each step.

Correction of GSIA

Any instance of GSIA terminates and compute σ^*, τ^*

The Hoffman Karp Algorithm

- the initial strategy: anything
- the set A of fixed arcs: the arcs going out of MAX-vertices
- how to find σ': solve the game $G[A, \sigma]$ (one player without randomness)

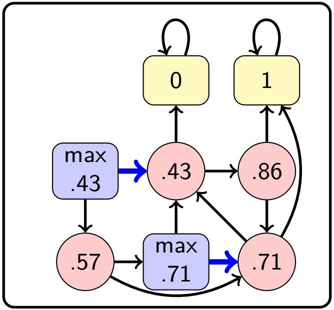
Algorithm in $O(2^n/n)$ iterations, lower bound in $2^{O(n)}$.

Changing the strategy on the upper left MAX-vertex is a **switch** and it increases the value.

The Hoffman Karp Algorithm

- the initial strategy: anything
- the set A of fixed arcs: the arcs going out of MAX-vertices
- how to find σ': solve the game $G[A, \sigma]$ (one player without randomness)

Algorithm in $O(2^n/n)$ iterations, lower bound in $2^{O(n)}$.



Changing the strategy on the upper left MAX-vertex is a **switch** and it increases the value.

One algorithm to rule them all

Why introducing yet another strategy improvement algorithm?

- capture all known SIAs
- relax the stopping condition (all plays stop almost surely)
- get better complexity bounds, function of the number of random vertices
- suggest new algorithms

Proof of correction

Two points in order to prove the correction of GSIA.

- If σ is not optimal in G, then σ is not optimal in $G[A, \sigma]$
- If $\sigma' \underset{G[A,\sigma]}{\succ} \sigma$ in $G[A, \sigma]$ then $\sigma' > \sigma$ in G

Equivalency of the value vector

The value of the vertices of $G[A, \sigma]$ under $(\sigma, \tau(\sigma))$ are the same as the value of G under $(\sigma, \tau(\sigma))$

Concatenated strategies

Concatenation

The strategy $\sigma'|_A\sigma$ plays as in σ' until the token reaches A then as in σ.

The strategy $\sigma'|_A\sigma$ is *not positional*.

Interpretation in the transformed game

For two MAX strategies σ, σ' and a subset of arcs A, we have:
$v^G_{\sigma'|_A\sigma} = v^{G[A,\sigma]}_{\sigma'}$.

Limits of concatenated strategies

$$\sigma'|_A^1\sigma \equiv \sigma'|_A\sigma$$

$$\sigma'|_A^{i+1}\sigma \equiv \sigma'|_A\sigma'|_A^i\sigma$$

Non decreasing sequence

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 $\sigma' \underset{G}{>} \sigma$.

Main ideas of the proof:

- order $\succ$ implies that there are *less* vertices in cycles going from σ to σ'
- induction to prove that $(\sigma'|_A^i\sigma)_i$ is increasing (in G), relying on the monotonicity of SSGs with regard to their sinks values
- the limit of $v^G_{\sigma'|^i\sigma}$ is $v^G_{\sigma'}$

Complexity of GSIA

- Function of n (binary SSGs), upper bound from the number of strategies: 2^n. Lower bound in $2^n - 1$.
- Function of r (binary SSGs), there is an improvement of a least 2^{-r} on a MAX-vertex in an iteration. At most $n2^r$ iterations and a lower bound of 2^{r+1}.
- For q-SSGs, nq^r iterations. When q is large, a $r!$ bound as for Gimbert and Horn's algorithm is better.
- Iterations which do not change the order of random vertices are cheap. Only rq^r heavy iterations.

Opt-GSIA is a restriction of GSIA, where A is always the same and σ' is the optimal strategy in $G[A, \sigma]$.
Open Question: Can we prove better bounds for Opt-GSIA?

Complexity of GSIA

- Function of n (binary SSGs), upper bound from the number of strategies: 2^n. Lower bound in $2^n - 1$.
- Function of r (binary SSGs), there is an improvement of a least 2^{-r} on a MAX-vertex in an iteration. At most $n2^r$ iterations and a lower bound of 2^{r+1}.
- For q-SSGs, nq^r iterations. When q is large, a $r!$ bound as for Gimbert and Horn's algorithm is better.
- Iterations which do not change the order of random vertices are cheap. Only rq^r heavy iterations.

Opt-GSIA is a restriction of GSIA, where A is always the same and σ' is the optimal strategy in $G[A, \sigma]$.
Open Question: Can we prove better bounds for Opt-GSIA?

f-strategies

f-strategies

Let f be a total ordering on $V_R \cup V_S$, $f : x_1 < x_2 < \cdots < x_{r+s}$. An f-strategy is an optimal strategy in the game where the $s + r$ vertices above are replaced by sinks with new values satisfying $\mathsf{Val}(x_1) < \mathsf{Val}(x_2) < \cdots < \mathsf{Val}(x_{r+s})$.

The order defines the strategy

An f-strategy does not depend on the value chosen for the sink and can be computed from f in linear time.

Gimbert and Horn propose to list all f-strategies or an SIA which transforms an f-strategy by updating the order to match the values at each step.

Complexity of generalized Gimbert and Horn's algorithm

Generalized GHA

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.

Comparison with the state of the art

Opt-GSIA, with A subset of the arcs going out of random vertices, needs less iterations than Ibsen-Jensen and Miltersen's algorithm to find the optimal values on any input.

- The speedup is exponential on some instances.
- When an arc out of each random vertex is in A, the transformed game is easy to solve.

New Algorithms

We consider instances of Opt-GSIA:

- the transformed game must be simple enough to be solvable in polynomial time
- the transformed game must be complex enough, so that finding its optimal solution improves values fast
- A is the set of arcs out of MIN vertices: single player transformed game (converge from below)

New Algorithms

We consider instances of Opt-GSIA:

- the transformed game must be simple enough to be solvable in polynomial time
- the transformed game must be complex enough, so that finding its optimal solution improves values fast
- A is the set of arcs out of MIN vertices: single player transformed game (converge from below)
- A is a feedback vertex set: acyclic transformed game

New Algorithms

We consider instances of Opt-GSIA:

- the transformed game must be simple enough to be solvable in polynomial time
- the transformed game must be complex enough, so that finding its optimal solution improves values fast
- A is the set of arcs out of MIN vertices: single player transformed game (converge from below)
- A is a feedback vertex set: acyclic transformed game

Thank you for your attention