David Auger, Pierre Coucheney, **Yann Strozecki**
Université de Versailles Saint-Quentin-en-Yvelines

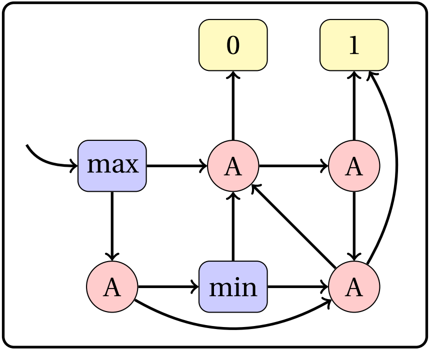
Almost Acyclic Simple Stochastic Games

Mardi 17 Décembre,
Journées CMF

Simple stochastic game (SSG)

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

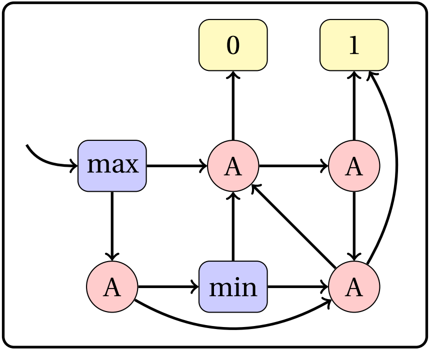
- three sets of vertices V_{MAX}, V_{MIN}, V_{AVE} of outdegree 2
- two (or more) 'sink' vertices with rational values



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 rational values

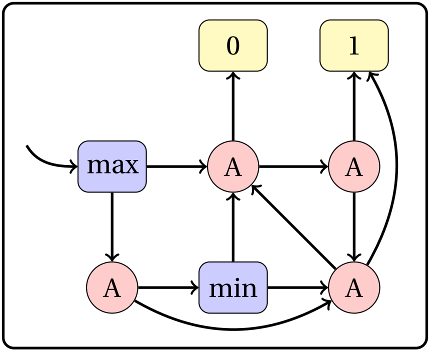


Two players : MAX and MIN, and randomness.

Rules of a SSG

A play consists in moving a pebble on the graph :

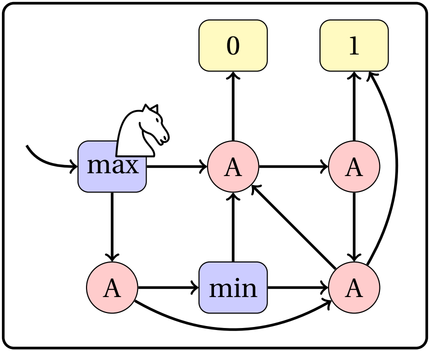
- player MAX wants to maximize the value of the sink reached.
- player MIN wants to minimize the value. If no sink is reached, the value is 0.



Rules of a 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.

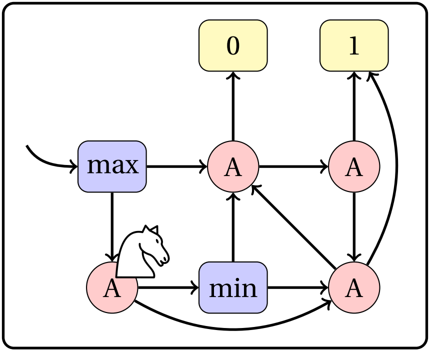


On a MAX node player MAX decides where to go next.

Rules of a 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.

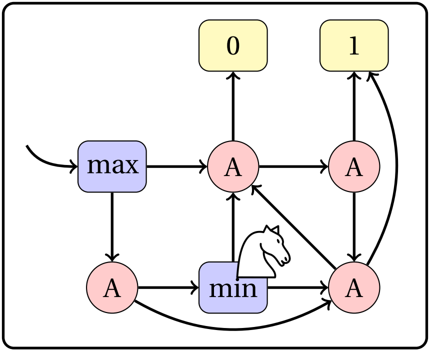


On a AVE node the next vertex is randomly determined.

Rules of a 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.

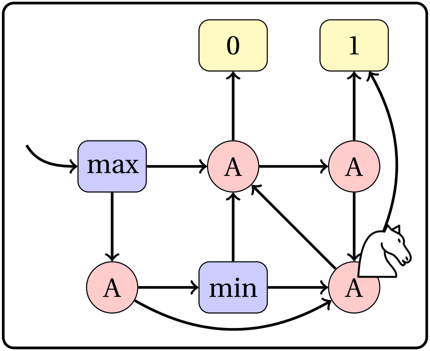


On a MIN node player MIN decides where to go next.

Rules of a 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.

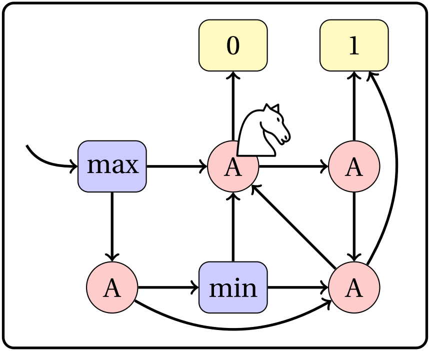


Etc.

Rules of a 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.

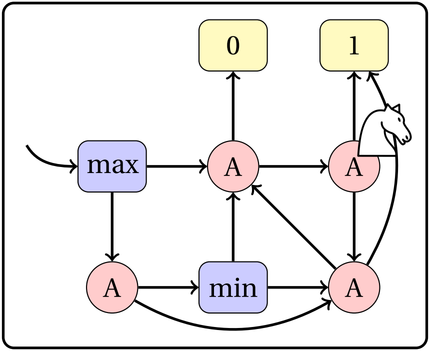


Etc.

Rules of a 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.

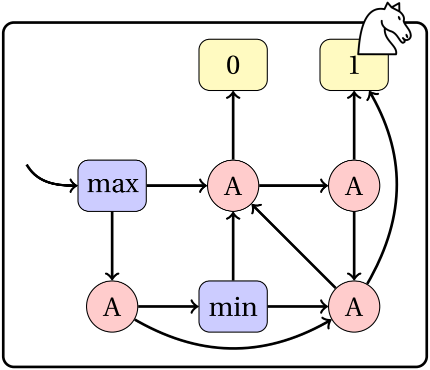


Etc.

Rules of a 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.

Strategies and values

General definition of a **strategy** σ for a player MAX :

$$\sigma : \text{partial play ending in } V_{\text{MAX}} \longmapsto \text{probability distribution on outneighbours}$$

Strategies and values

General definition of a **strategy** σ for a player MAX :

$$\sigma : \text{partial play ending in } V_{\text{MAX}} \longmapsto \text{probability distribution on outneighbours}$$

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

$$v(x) = \sup_{\substack{\sigma \text{ strategy} \\ \text{for MAX}}} \inf_{\substack{\tau \text{ strategy} \\ \text{for MIN}}} \underbrace{\mathbb{E}_{\sigma,\tau}\left(\text{value of the sink reached} \mid \text{game starts in } x\right)}_{v_{\sigma,\tau}(x)}$$

Strategies and values

General definition of a **strategy** σ for a player MAX :

$$\sigma : \text{partial play ending in } V_{\mathrm{MAX}} \longmapsto \text{probability distribution on outneighbours}$$

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

$$v(x) = \sup_{\substack{\sigma \text{ strategy} \\ \text{for MAX}}} \; \inf_{\substack{\tau \text{ strategy} \\ \text{for MIN}}} \; \underbrace{\mathbb{E}_{\sigma,\tau}\left(\text{value of the sink reached} \mid \text{game starts in } x\right)}_{v_{\sigma,\tau}(x)}$$

<u>**Problem :**</u> given a game and a vertex, compute the value of the vertex.

Strategies and values

General definition of a **strategy** σ for a player MAX :

$$\sigma : \text{partial play ending in } V_{\text{MAX}} \longmapsto \text{probability distribution on outneighbours}$$

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

$$v(x) = \sup_{\substack{\sigma \text{ strategy} \\ \text{for MAX}}} \inf_{\substack{\tau \text{ strategy} \\ \text{for MIN}}} \underbrace{\mathbb{E}_{\sigma,\tau}\left(\text{value of the sink reached} \mid \text{game starts in } x\right)}_{v_{\sigma,\tau}(x)}$$

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

Decision problem : $v(x) > 0.5$?

Why simple stochastic games ?

They generalize important models such as :

- Parity games (model checking of μ-calculus)
- Mean payoff games (useful for optimisation)
- Linear programming
- Markov decision process

Why simple stochastic games ?

They generalize important models such as :

- Parity games (model checking of μ-calculus)
- Mean payoff games (useful for optimisation)
- Linear programming
- Markov decision process

Also there are :

- An example of a problem yet between P and NP
- A simple framework to study stochastic games
- A good model to study partial information

Simpler game : Stopping SSGs

A SSG is stopping if for all strategies, the game reaches a sink vertex almost surely.

Simpler game : Stopping SSGs

A SSG is stopping if for all strategies, the game reaches a sink vertex almost surely.

Theorem (Condon 89)

For every SSG G, there is a polynomial-time computable SSG G' such that

- *G' is stopping*
- *size of G' = poly(size of G)*
- *for all vertices x,* $v_{G'}(x) > \frac{1}{2}$ *if and only if* $v_G(x) > \frac{1}{2}$

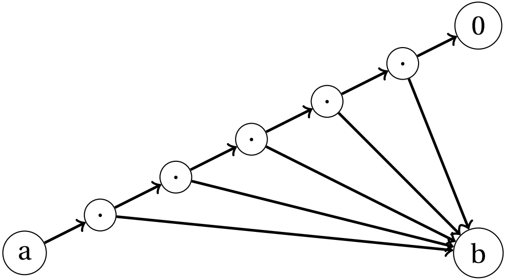
Simpler game : Stopping SSGs

A SSG is stopping if for all strategies, the game reaches a sink vertex almost surely.

Theorem (Condon 89)

For every SSG G, there is a polynomial-time computable SSG G' such that

- *G' is stopping*
- *size of G' = poly(size of G)*
- *for all vertices x, $v_{G'}(x) > \frac{1}{2}$ if and only if $v_G(x) > \frac{1}{2}$*



Simpler strategies

To compute values we can restrict our strategies to be

- pure : deterministic
- memoryless : does not depend on the entire history
- stationary : does not depend on time step

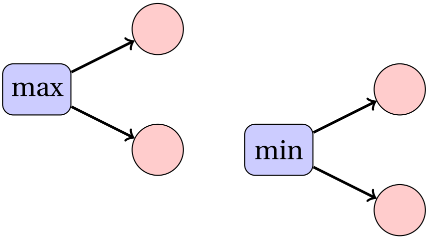
Simpler strategies

To compute values we can restrict our strategies to be

- pure : deterministic
- memoryless : does not depend on the entire history
- stationary : does not depend on time step

We call them **positional strategies** for short.

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



Optimality of positional strategies

Lemma

Against a **positional** *strategy* σ, MIN *might as well respond positional :*

$$\sigma \textit{ positional} \Rightarrow \quad \inf_{\tau \textit{ general}} v_{\sigma,\tau}(x) = \min_{\tau \textit{ positional}} v_{\sigma,\tau}(x)$$

Optimality of positional strategies

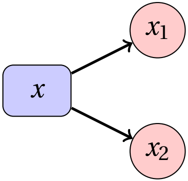
Lemma

Against a **positional** *strategy* σ, MIN *might as well respond positional :*

$$\sigma \textit{ positional} \Rightarrow \inf_{\tau \textit{ general}} v_{\sigma,\tau}(x) = \min_{\tau \textit{ positional}} v_{\sigma,\tau}(x)$$

Bellman equation characterizes optimality :

$$v^* = \min_{\tau \text{ general}} v_{\sigma,\tau} \Leftrightarrow \forall x, v^*(x) = \begin{cases} \min(v^*(x_1), v^*(x_2)) & \text{if } x \in \text{MIN} \\ \frac{1}{2}(v^*(x_1) + v^*(x_2)) & \text{if } x \in \text{AVE} \\ 0 \text{ or } 1 & \text{if } x \in \text{SINK} \end{cases}$$



Optimality of positional strategies

Lemma

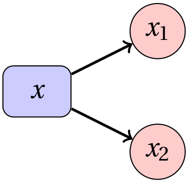
Against a **positional** *strategy* σ, MIN *might as well respond positional :*

$$\sigma \textit{ positional} \Rightarrow \quad \inf_{\tau \textit{ general}} v_{\sigma,\tau}(x) = \min_{\tau \textit{ positional}} v_{\sigma,\tau}(x)$$

Bellman equation characterizes optimality :

$$v^* = \min_{\tau \text{ general}} v_{\sigma,\tau} \Leftrightarrow \forall x, v^*(x) = \begin{cases} \min(v^*(x_1), v^*(x_2)) & \text{if } x \in \text{MIN} \\ \frac{1}{2}(v^*(x_1) + v^*(x_2)) & \text{if } x \in \text{AVE} \\ 0 \text{ or } 1 & \text{if } x \in \text{SINK} \end{cases}$$

A solution exists and is unique.



Optimality of positional strategies

Lemma

Against a **positional** *strategy* σ, MIN *might as well respond positional :*

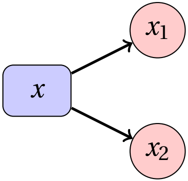
$$\sigma \textit{ positional} \Rightarrow \quad \inf_{\tau \textit{ general}} v_{\sigma,\tau}(x) = \min_{\tau \textit{ positional}} v_{\sigma,\tau}(x)$$

Bellman equation characterizes optimality :

$$v^* = \min_{\tau \text{ general}} v_{\sigma,\tau} \Leftrightarrow \forall x, v^*(x) = \begin{cases} \min(v^*(x_1), v^*(x_2)) & \text{if } x \in \text{MIN} \\ \frac{1}{2}(v^*(x_1) + v^*(x_2)) & \text{if } x \in \text{AVE} \\ 0 \text{ or } 1 & \text{if } x \in \text{SINK} \end{cases}$$

A solution exists and is unique.

An optimal positional strategy consists in playing optimally at each node wrt to v^*.



Minimax Theorem

Theorem (Condon 89)

For all vertices x,

$$
\begin{aligned}
v(x) &= \sup_{\sigma \text{ general}} \inf_{\tau \text{ general}} v_{\sigma,\tau}(x) \\
&= \inf_{\tau \text{ general}} \sup_{\sigma \text{ general}} v_{\sigma,\tau}(x) \\
&= \max_{\sigma \text{ positional}} \min_{\tau \text{ positional}} v_{\sigma,\tau}(x) \\
&= \min_{\tau \text{ positional}} \max_{\sigma \text{ positional}} v_{\sigma,\tau}(x)
\end{aligned}
$$

Complexity Considerations

Theorem (Condon 89)

The SSG value problem is in $\mathsf{NP} \cap \mathsf{coNP}$.

Because of the optimality conditions and the symmetry between MAX and MIN.

Complexity Considerations

Theorem (Condon 89)

The SSG value problem is in $\mathsf{NP} \cap \mathsf{coNP}$.

Because of the <u>optimality conditions</u> and the <u>symmetry</u> between MAX and MIN.

Lemma

Stopping game hypothesis $\Rightarrow$ *unique pair of optimal strategies.*

Complexity Considerations

Theorem (Condon 89)

The SSG value problem is in $\mathsf{NP} \cap \mathsf{coNP}$.

Because of the optimality conditions and the symmetry between MAX and MIN.

Lemma

Stopping game hypothesis $\Rightarrow$ *unique pair of optimal strategies.*

The problem is in $\mathsf{UP} \cap \mathsf{coUP}$ (unique certificate).

Complexity Considerations

Theorem (Condon 89)

The SSG value problem is in $\mathsf{NP} \cap \mathsf{coNP}$.

Because of the optimality conditions and the symmetry between MAX and MIN.

Lemma

Stopping game hypothesis $\Rightarrow$ *unique pair of optimal strategies.*

The problem is in $\mathsf{UP} \cap \mathsf{coUP}$ (unique certificate).

The problem is complete for logspace alternating randomized Turing machine or **game against nature**.

Complexity Considerations

Theorem (Condon 89)

The SSG value problem is in $\mathsf{NP} \cap \mathsf{coNP}$.

Because of the optimality conditions and the symmetry between MAX and MIN.

Lemma

Stopping game hypothesis $\Rightarrow$ *unique pair of optimal strategies.*

The problem is in $\mathsf{UP} \cap \mathsf{coUP}$ (unique certificate).

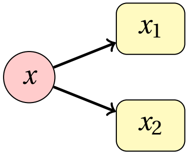
The problem is complete for logspace alternating randomized Turing machine or **game against nature**.

Open question : is the value problem in P ?

Computing values

Fix σ, τ positional strategies.

- if $x \in \text{MAX}$, $v_{\sigma,\tau}(x) = v_{\sigma,\tau}(\sigma(x))$
- if $x \in \text{MIN}$, $v_{\sigma,\tau}(x) = v_{\sigma,\tau}(\tau(x))$
- if $x \in \text{AVE}$, $v_{\sigma,\tau}(x) = \frac{1}{2} v_{\sigma,\tau}(x_1) + \frac{1}{2} v_{\sigma,\tau}(x_2)$
- if $x \in \text{SINK}$, $v_{\sigma,\tau}(x) \in [0, 1]$

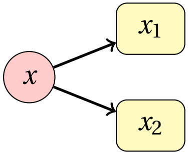


This amounts to solve a linear system.

Computing values

Fix σ only (best response).

- if $x \in \mathrm{MAX}$, $v_\sigma(x) = v_\sigma(\sigma(x))$
- if $x \in \mathrm{MIN}$, $v_\sigma(x) \leq v_\sigma(x_1)$ and $v_\sigma(x) \leq v_\sigma(x_2)$
- if $x \in \mathrm{AVE}$, $v_\sigma(x) = \frac{1}{2} v_{\sigma,}(x_1) + \frac{1}{2} v_\sigma(x_2)$
- if $x \in \mathrm{SINK}$, $v_\sigma(x) \in [0,1]$
- $\max \sum_{x \in \mathrm{MIN}} v_\sigma(x)$

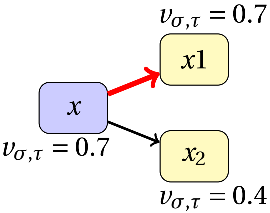


This amounts to solve a linear program.

The switch operation

x is a MIN vertex and $v_{\sigma,\tau}(x) = v_{\sigma,\tau}(x_1) > v_{\sigma,\tau}(x_2)$

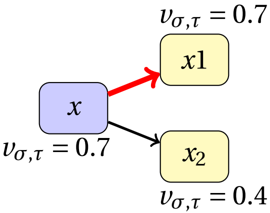
switching τ at $x : \tau'(x) = x_2$ and equal to τ elsewhere.



The switch operation

x is a MIN vertex and $v_{\sigma,\tau}(x) = v_{\sigma,\tau}(x_1) > v_{\sigma,\tau}(x_2)$

switching τ at x : $\tau'(x) = x_2$ and equal to τ elsewhere.



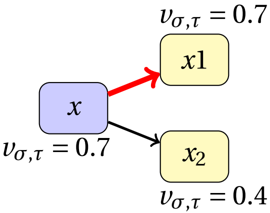
Such a switch is **profitable** for MIN :

- for all y, $v_{\sigma,\tau'}(y) \leq v_{\sigma,\tau}(y)$
- in particular $v_{\sigma,\tau'}(x) < v_{\sigma,\tau}(x)$

The switch operation

x is a MIN vertex and $v_{\sigma,\tau}(x) = v_{\sigma,\tau}(x_1) > v_{\sigma,\tau}(x_2)$

switching τ at x : $\tau'(x) = x_2$ and equal to τ elsewhere.



Such a switch is **profitable** for MIN :

- for all y, $v_{\sigma,\tau'}(y) \leq v_{\sigma,\tau}(y)$
- in particular $v_{\sigma,\tau'}(x) < v_{\sigma,\tau}(x)$

Optimality condition : no switch .
Algorithm to find an optimal strategy against σ : keep switching.

Strategy improvement algorithms

The strategy improvement algorithm a.k.a Hoffman-Karp algorithm (1966, MDP context) is

1. choose τ_0 and let $\sigma_0 = \sigma(\tau_0)$ (best response)
2. while (σ_k, τ_k) is not optimal, obtain τ_{k+1} by switching τ_k; let $\sigma_{k+1} = \sigma(\tau_{k+1})$

based on :

Lemma

$v_{\sigma_{k+1},\tau_{k+1}} < v_{\sigma_k,\tau_k}$ as long as (σ_k, τ_k) is not optimal.

Strategy improvement algorithms

The strategy improvement algorithm a.k.a Hoffman-Karp algorithm (1966, MDP context) is

1. choose τ_0 and let $\sigma_0 = \sigma(\tau_0)$ (best response)
2. while (σ_k, τ_k) is not optimal, obtain τ_{k+1} by switching τ_k; let $\sigma_{k+1} = \sigma(\tau_{k+1})$

based on :

Lemma

$v_{\sigma_{k+1},\tau_{k+1}} < v_{\sigma_k,\tau_k}$ as long as (σ_k, τ_k) is not optimal.

Theorem

The HK algorithm makes at most $O(2^n/n)$ iterations

Unfortunately, this can take exponential time [Condon, Friedman].

Strategy improvement algorithms

The strategy improvement algorithm a.k.a Hoffman-Karp algorithm (1966, MDP context) is

1. choose τ_0 and let $\sigma_0 = \sigma(\tau_0)$ (best response)
2. while (σ_k, τ_k) is not optimal, obtain τ_{k+1} by switching τ_k; let $\sigma_{k+1} = \sigma(\tau_{k+1})$

based on :

Lemma

$v_{\sigma_{k+1},\tau_{k+1}} < v_{\sigma_k,\tau_k}$ as long as (σ_k, τ_k) is not optimal.

Theorem

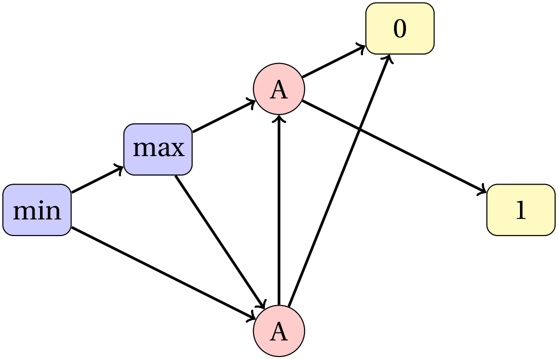
The HK algorithm makes at most $O(2^n/n)$ iterations

Unfortunately, this can take exponential time [Condon, Friedman].

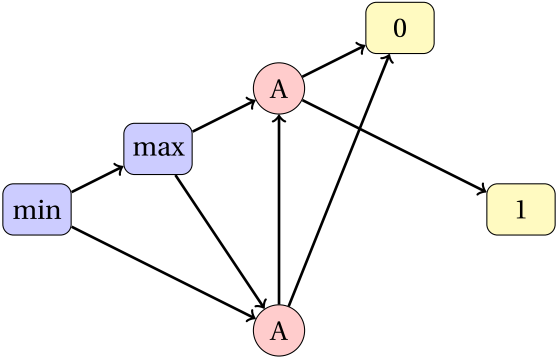
When the algorithm ends, say at (σ^*, τ^*), each one plays optimally :

$$v_{\sigma^*,\tau^*} = \max_{\sigma \text{ pos}} \min_{\tau \text{ pos}} v_{\sigma,\tau} = \min_{\tau \text{ pos}} \max_{\sigma \text{ pos}} v_{\sigma,\tau}$$

Solving an acyclic SSG in linear time



Solving an acyclic SSG in linear time



No cycle : compute the values backward from the sinks in time $O(n)$.

Milder form of acyclicity

Max-acyclic game : each MAX vertex has at most one outgoing edge in a cycle.

Milder form of acyclicity

Max-acyclic game : each MAX vertex has at most one outgoing edge in a cycle.

Assume the graph is strongly connected once sinks are removed $\Rightarrow$ one outneighbour of each MAX vertex is a sink.

Milder form of acyclicity

Max-acyclic game : each MAX vertex has at most one outgoing edge in a cycle.

Assume the graph is strongly connected once sinks are removed $\Rightarrow$ one outneighbour of each MAX vertex is a sink.

$x \in$MAX is **open**/**closed** : the strategy chooses a sink/not a sink.

Theorem

The strategy improvment algorithm (MAX switches + MIN responds optimally) starting with open MAX vertices performs at most $|V_{MAX}|$ switches.

Milder form of acyclicity

Max-acyclic game : each MAX vertex has at most one outgoing edge in a cycle.

Assume the graph is strongly connected once sinks are removed ⇒ one outneighbour of each MAX vertex is a sink.

$x \in$MAX is **open**/**closed** : the strategy chooses a sink/not a sink.

Theorem

The strategy improvment algorithm (MAX switches + MIN responds optimally) starting with open MAX vertices performs at most $|V_{MAX}|$ switches.

proof : once a MAX vertex is closed, it is for ever.

Milder form of acyclicity

Max-acyclic game : each MAX vertex has at most one outgoing edge in a cycle.

Assume the graph is strongly connected once sinks are removed $\Rightarrow$ one outneighbour of each MAX vertex is a sink.

$x \in$MAX is **open**/**closed** : the strategy chooses a sink/not a sink.

Theorem

The strategy improvment algorithm (MAX switches + MIN responds optimally) starting with open MAX vertices performs at most $|V_{MAX}|$ switches.

proof : once a MAX vertex is closed, it is for ever.

Need to compute values $\Rightarrow O(n^4|V_{MAX}|)$.

1-cycle SSG

With only one cycle the game is both Max-acyclic and Min-acyclic.

1-cycle SSG

With only one cycle the game is both Max-acyclic and Min-acyclic.

In linear time, each of these assumptions can be checked : the optimal strategy

A1 is closed at every vertex

A2 contains at least an open MAX vertex

A3 contains at least an open MIN vertex

1-cycle SSG

With only one cycle the game is both Max-acyclic and Min-acyclic.

In linear time, each of these assumptions can be checked : the optimal strategy

A1 is closed at every vertex

A2 contains at least an open MAX vertex

A3 contains at least an open MIN vertex

algo for checking **A2** :

1. solve the acyclic SSG obtained when an arbitrary MAX vertex is fixed to open
2. if **A2** is true, the next open MAX vertex (say x) is also open in the 1-cycle SSG
3. so check it by solving the acyclic SSG forced to be open at x.

k-cycles SSG

Based on the previous algo, by opening a vertex next to each fork vertex :

Theorem

A k-cycles SSG can be solved in time $O(nk!)$

k-cycles SSG

Based on the previous algo, by opening a vertex next to each fork vertex :

Theorem

A k-cycles SSG can be solved in time $O(nk!)$

To be compared to the strategy improvment algorithm : $O(n^4 2^k)$

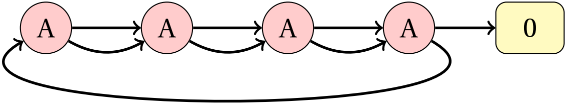
SSG with feedback vertex set of size k

There are k vertices that, once removed, yields an acyclic SSG.

SSG with feedback vertex set of size k

There are k vertices that, once removed, yields an acyclic SSG.

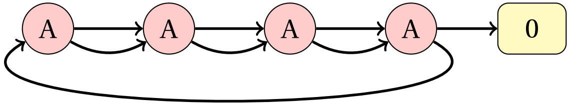
⚠ even if $k = 1$, the number of cycles can be large



SSG with feedback vertex set of size k

There are k vertices that, once removed, yields an acyclic SSG.

⚠ even if $k = 1$, the number of cycles can be large



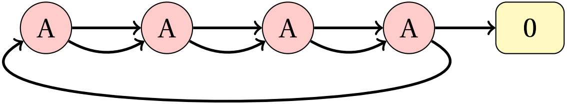
Bisection algo with $k = 1$: x is the vertex to remove to get an acyclic SSG

1. solve the acyclic SSG obtained when x is replaced by a sink with value $s = \frac{min+max}{2}$

SSG with feedback vertex set of size k

There are k vertices that, once removed, yields an acyclic SSG.

⚠ even if $k = 1$, the number of cycles can be large



Bisection algo with $k = 1$: x is the vertex to remove to get an acyclic SSG

1. solve the acyclic SSG obtained when x is replaced by a sink with value $s = \frac{min+max}{2}$
2. if value s satisfies the local optimality condition (up to some error bound)

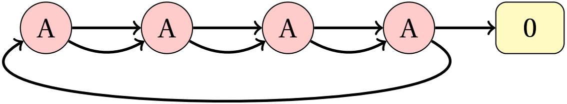
$$s \in [\min(v(x_1), v(x_2)) - \epsilon\,;\, \min(v(x_1), v(x_2)) + \epsilon]$$

then s is close to the real value of x in the initial game

SSG with feedback vertex set of size k

There are k vertices that, once removed, yields an acyclic SSG.

⚠ even if $k = 1$, the number of cycles can be large



Bisection algo with $k = 1$: x is the vertex to remove to get an acyclic SSG

1. solve the acyclic SSG obtained when x is replaced by a sink with value $s = \frac{min+max}{2}$
2. if value s satisfies the local optimality condition (up to some error bound)

$$s \in [\min(v(x_1), v(x_2)) - \epsilon \,;\, \min(v(x_1), v(x_2)) + \epsilon]$$

   then s is close to the real value of x in the initial game
3. otherwise if $s > \min(v(x_1), v(x_2)) + \epsilon \Rightarrow$ then the value of x is less than s, go back to 1 with $max = s$.

SSG with feedback vertex set of size k

We can use the bisection algorithm on a k-dimensional space.

SSG with feedback vertex set of size k

We can use the bisection algorithm on a k-dimensional space.

⚠ Problem with the precision of the values → exact computation.

SSG with feedback vertex set of size k

We can use the bisection algorithm on a k-dimensional space.

⚠ Problem with the precision of the values → exact computation.

Theorem

A SSG with feedback vertex set of size k can be solved in time $O(n^{k+1})$

SSG with feedback vertex set of size k

We can use the bisection algorithm on a k-dimensional space.

⚠ Problem with the precision of the values → exact computation.

Theorem

A SSG with feedback vertex set of size k can be solved in time $O(n^{k+1})$

The method can be used to remove k vertices in any SSG and thus makes other classes of SSG tractable.

Tanks for listening !

Questions ?