Simple stochastic games: a state of the art

Yann Strozecki

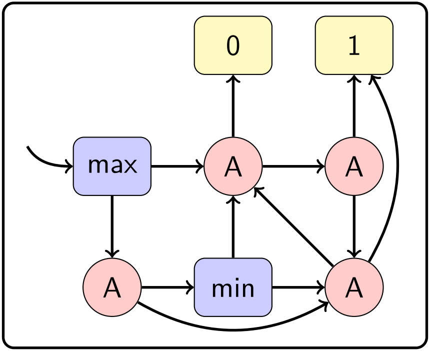
Université de Versailles St-Quentin-en-Yvelines
Laboratoire PRiSM

June 2013, Gödel Research Center

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 'sink' vertices 0 and 1

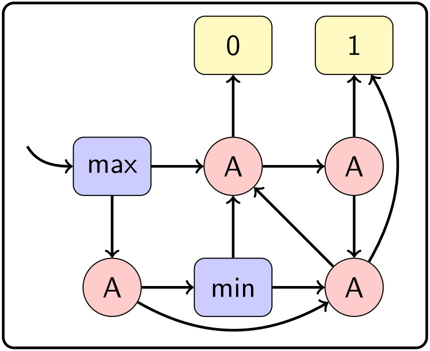


Two players: MAX and MIN, and *randomness*.

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 'sink' vertices 0 and 1

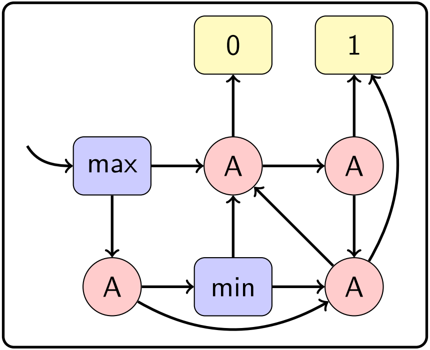


Two players: MAX and MIN, and *randomness*.

Semantic of SSGs

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

- player MAX wants to reach the 1 sink
- player MIN wants to prevent him from doing so

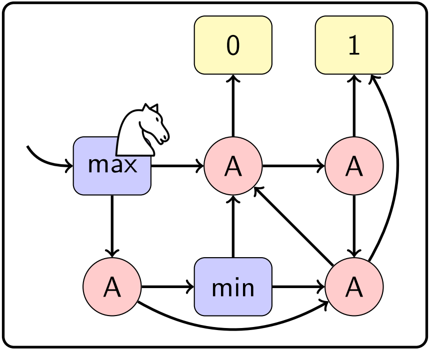


On a MAX node player MAX decides where to go next.

Semantic of SSGs

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

- player MAX wants to reach the 1 sink
- player MIN wants to prevent him from doing so

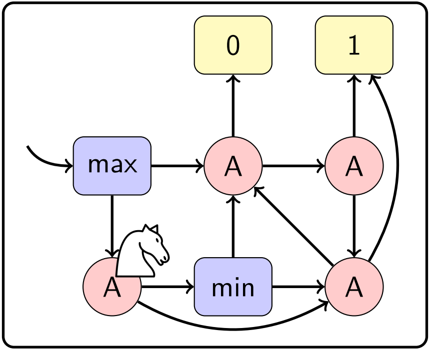


On a MAX node player MAX decides where to go next.

Semantic of SSGs

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

- player MAX wants to reach the 1 sink
- player MIN wants to prevent him from doing so

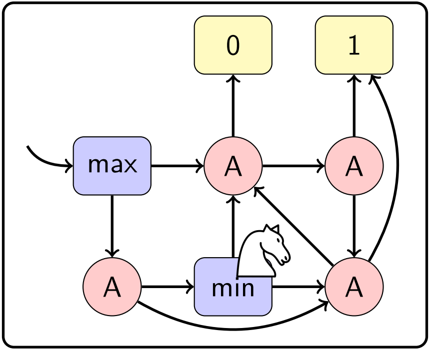


On a AVE node the next vertex is randomly determined.

Semantic of SSGs

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

- player MAX wants to reach the 1 sink
- player MIN wants to prevent him from doing so

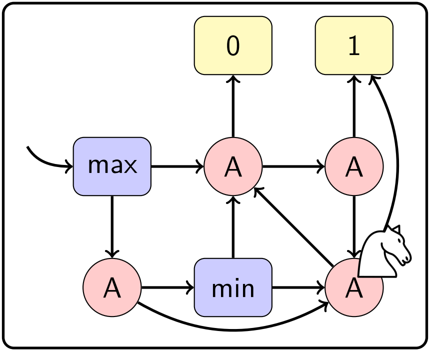


On a MIN node player MIN decides where to go next.

Semantic of SSGs

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

- player MAX wants to reach the 1 sink
- player MIN wants to prevent him from doing so

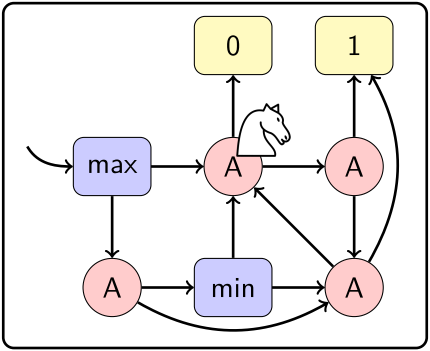


Etc.

Semantic of SSGs

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

- player MAX wants to reach the 1 sink
- player MIN wants to prevent him from doing so

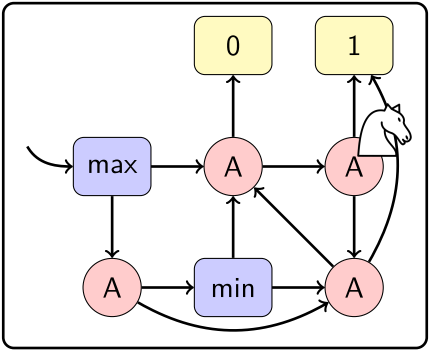


Etc.

Semantic of SSGs

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

- player MAX wants to reach the 1 sink
- player MIN wants to prevent him from doing so

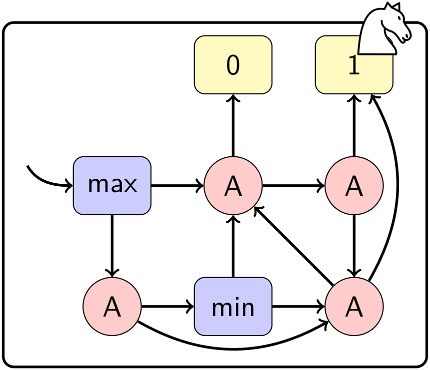


Etc.

Semantic of SSGs

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

- player MAX wants to reach the 1 sink
- player MIN wants to prevent him from doing so



Etc.

Strategies and values

General definition of a strategy σ for a player MAX:

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

The value of a vertex x is:

$$v(x) = \sup_{\substack{\sigma \text{ strategy} \\ \text{for } \mathrm{MAX}}} \underbrace{\inf_{\substack{\tau \text{ strategy} \\ \text{for } \mathrm{MIN}}} \underbrace{\mathbb{P}_{\sigma,\tau}\left(1 \text{ is reached} \;\middle|\; \text{game starts in } x\right)}_{v_{\sigma,\tau}(x)}}_{v_\sigma(x)}$$

Strategies and values

General definition of a strategy σ for a player MAX:

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

The value of a vertex x is:

$$v(x) = \sup_{\substack{\sigma \text{ strategy} \\ \text{for } \mathrm{MAX}}} \underbrace{\inf_{\substack{\tau \text{ strategy} \\ \text{for } \mathrm{MIN}}} \underbrace{\mathbb{P}_{\sigma,\tau}\left(1 \text{ is reached} \mid \text{game starts in } x\right)}_{v_{\sigma,\tau}(x)}}_{v_\sigma(x)}$$

Problem: 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:

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

The value of a vertex x is:

$$v(x) = \sup_{\substack{\sigma \text{ strategy} \\ \text{for } \mathrm{MAX}}} \underbrace{\inf_{\substack{\tau \text{ strategy} \\ \text{for } \mathrm{MIN}}} \underbrace{\mathbb{P}_{\sigma,\tau}\left(1 \text{ is reached} \;\middle|\; \text{game starts in } x\right)}_{v_{\sigma,\tau}(x)}}_{v_{\sigma}(x)}$$

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

Discounted payoff games

Played on a graph as SSGs, but no sink vertices nor average vertices.

To each edge is associated a payoff w and a discount factor λ.

Discounted payoff games

Played on a graph as SSGs, but no sink vertices nor average vertices.

To each edge is associated a payoff w and a discount factor λ.

The value of a play $e_{i_0} e_{i_1} e_{i_2} \ldots$ is $\sum_{j=0}^{\infty} w(e_{i_j}) \prod_{k<j} \lambda_{i_k}$.

Discounted payoff games

Played on a graph as SSGs, but no sink vertices nor average vertices.

To each edge is associated a payoff w and a discount factor λ.

The value of a play $e_{i_0} e_{i_1} e_{i_2} \dots$ is $\sum_{j=0}^{\infty} w(e_{i_j}) \prod_{k<j} \lambda_{i_k}$.

Theorem

There is a reduction from DPG to SSG, such that a vertex has payoff 1 in the DPG if the corresponding vertex has value $> \frac{1}{2}$ in the SSG

Discounted payoff games

Played on a graph as SSGs, but no sink vertices nor average vertices.

To each edge is associated a payoff w and a discount factor λ.

The value of a play $e_{i_0} e_{i_1} e_{i_2} \dots$ is $\sum_{j=0}^{\infty} w(e_{i_j}) \prod_{k<j} \lambda_{i_k}$.

Theorem

There is a reduction from DPG to SSG, such that a vertex has payoff 1 in the DPG if the corresponding vertex has value $> \frac{1}{2}$ in the SSG

Idea: each vertex from the DPG has a probability to go to the sinks chosen to simulate the reward and the discount factor.

Discounted payoff games

Played on a graph as SSGs, but no sink vertices nor average vertices.

To each edge is associated a payoff w and a discount factor λ.

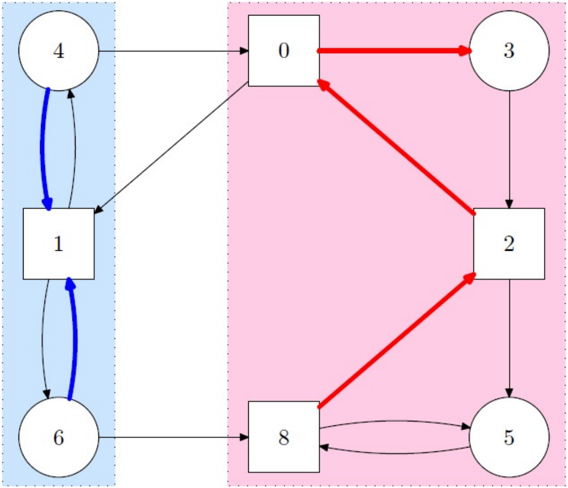
The value of a play $e_{i_0}\, e_{i_1}\, e_{i_2} \ldots$ is $\sum_{j=0}^{\infty} w(e_{i_j}) \prod_{k<j} \lambda_{i_k}$.

Theorem

There is a reduction from DPG to SSG, such that a vertex has payoff 1 in the DPG if the corresponding vertex has value $> \frac{1}{2}$ in the SSG

Idea: each vertex from the DPG has a probability to go to the sinks chosen to simulate the reward and the discount factor.

Parity games



- Two player game on a graph
- Play goes on forever
- Every vertex has a priority
- P set of infinitely seen priority

If the largest value of P is even, player 0 wins otherwise 1 wins.

Reduction from Parity games to SSGs

Theorem

There is a reduction from parity games to simple stochastic games, such that a vertex is winning for 1 in the PG if the corresponding vertex has value $> \frac{1}{2}$ in the SSG

Idea:

- Add two sinks 0 and 1.
- Assign for every transition a small probability to go to sink 0 (nodes of player 0) or sink 1 (nodes of player 1).
- The transition probability from a node of priority i must be superior to the sum of transition probabilities of the nodes of priority less than i.

Reduction from Parity games to SSGs

Theorem

There is a reduction from parity games to simple stochastic games, such that a vertex is winning for 1 in the PG if the corresponding vertex has value $> \frac{1}{2}$ in the SSG

Idea:

- Add two sinks 0 and 1.
- Assign for every transition a small probability to go to sink 0 (nodes of player 0) or sink 1 (nodes of player 1).
- The transition probability from a node of priority i must be superior to the sum of transition probabilities of the nodes of priority less than i.

Complexity basics

A language: $L \subseteq \Sigma^*$

A problem: given an instance $w \in \Sigma^*$ decide whether $w \in L$.

Complexity basics

A language: $L \subseteq \Sigma^*$

A problem: given an instance $w \in \Sigma^*$ decide whether $w \in L$.

The SSG value problem is *given a SSG and one of its vertices x, is $v(x) > \frac{1}{2}$?*

Complexity basics

A language: $L \subseteq \Sigma^*$

A problem: given an instance $w \in \Sigma^*$ decide whether $w \in L$.

The SSG value problem is *given a SSG and one of its vertices x, is $v(x) > \frac{1}{2}$?*

Complexity of a problem: time taken by a Turing machine to solve the problem with regard to the size of the instance.

Complexity basics

A language: $L \subseteq \Sigma^*$

A problem: given an instance $w \in \Sigma^*$ decide whether $w \in L$.

The SSG value problem is *given a SSG and one of its vertices x, is $v(x) > \frac{1}{2}$?*

Complexity of a problem: time taken by a Turing machine to solve the problem with regard to the size of the instance.

A time polynomial in the size of the instance: the language is in P

Complexity basics

A language: $L \subseteq \Sigma^*$

A problem: given an instance $w \in \Sigma^*$ decide whether $w \in L$.

The SSG value problem is *given a SSG and one of its vertices x, is $v(x) > \frac{1}{2}$?*

Complexity of a problem: time taken by a Turing machine to solve the problem with regard to the size of the instance.

A time polynomial in the size of the instance: the language is in P

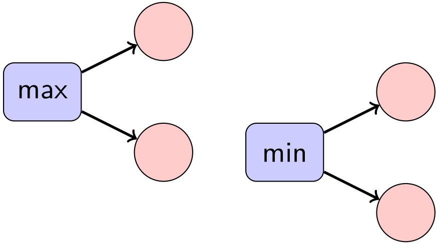
Simpler strategies

To compute values we can restrict our strategies to be

- *pure*: deterministic
- *memoryless*: does not depend from the memory

We call them positional strategies for short.

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



Forgetting the past makes it simpler

Pure:
Let σ be a randomized strategy which on vertex a chooses with probability λ the vertex b and with probability $1 - \lambda$ the vertex c.

The value of strategy σ,
$v_\sigma(a) = \lambda v_\sigma(b) + (1 - \lambda) v_\sigma(c)$

Forgetting the past makes it simpler

Pure:
Let σ be a randomized strategy which on vertex a chooses with probability λ the vertex b and with probability $1-\lambda$ the vertex c.

The value of strategy σ,
$v_\sigma(a) = \lambda v_\sigma(b) + (1-\lambda) v_\sigma(c)$

Say that $v_\sigma(b) > v_\sigma(c)$. The strategy σ' which always chooses b is better than σ.

Forgetting the past makes it simpler

Pure:
Let σ be a randomized strategy which on vertex a chooses with probability λ the vertex b and with probability $1-\lambda$ the vertex c.

The value of strategy σ,
$v_\sigma(a) = \lambda v_\sigma(b) + (1-\lambda)v_\sigma(c)$

Say that $v_\sigma(b) > v_\sigma(c)$. The strategy σ' which always chooses b is better than σ.

Memoryless:
Assume you have an optimal strategy which depends on the memory.

Forgetting the past makes it simpler

Pure:
Let σ be a randomized strategy which on vertex a chooses with probability λ the vertex b and with probability $1-\lambda$ the vertex c.

The value of strategy σ,
$v_\sigma(a) = \lambda v_\sigma(b) + (1-\lambda)v_\sigma(c)$

Say that $v_\sigma(b) > v_\sigma(c)$. The strategy σ' which always chooses b is better than σ.

Memoryless:
Assume you have an optimal strategy which depends on the memory.

When the pebble is at vertex v after a sequence of move, play as the optimal strategy assuming v is the starting vertex.

Forgetting the past makes it simpler

Pure:
Let σ be a randomized strategy which on vertex a chooses with probability λ the vertex b and with probability $1 - \lambda$ the vertex c.

The value of strategy σ,
$v_\sigma(a) = \lambda v_\sigma(b) + (1 - \lambda) v_\sigma(c)$

Say that $v_\sigma(b) > v_\sigma(c)$. The strategy σ' which always chooses b is better than σ.

Memoryless:
Assume you have an optimal strategy which depends on the memory.

When the pebble is at vertex v after a sequence of move, play as the optimal strategy assuming v is the starting vertex.

The number of optimal strategies is bounded: the value problem is decidable.

Forgetting the past makes it simpler

Pure:
Let σ be a randomized strategy which on vertex a chooses with probability λ the vertex b and with probability $1 - \lambda$ the vertex c.

The value of strategy σ,
$v_\sigma(a) = \lambda v_\sigma(b) + (1 - \lambda) v_\sigma(c)$

Say that $v_\sigma(b) > v_\sigma(c)$. The strategy σ' which always chooses b is better than σ.

Memoryless:
Assume you have an optimal strategy which depends on the memory.

When the pebble is at vertex v after a sequence of move, play as the optimal strategy assuming v is the starting vertex.

The number of optimal strategies is bounded: the value problem is decidable.

Minimax Theorem

Theorem (Condon 89)

For all vertices x,

$$\begin{aligned} v(x) &= \max_{\substack{\sigma \text{ positional strategy} \\ \text{for MAX}}} \; \min_{\substack{\tau \text{ positional strategy} \\ \text{for MIN}}} \; v_{\sigma,\tau}(x) \\ &= \min_{\substack{\tau \text{ positional strategy} \\ \text{for MIN}}} \; \max_{\substack{\sigma \text{ positional strategy} \\ \text{for MAX}}} \; v_{\sigma,\tau}(x) \end{aligned}$$

Main lines of a proof ...

1. Sups and infs are maxs and mins: optimal strategies and best responses exists (compacity and continuity arguments)
2. Against a positional strategy σ, MIN might as well respond positional:

$$\min_{\tau \text{ general}} v_{\sigma,\tau}(x) = \min_{\tau \text{ positional}} v_{\sigma,\tau}(x)$$

Main lines of a proof ...

1. Sups and infs are maxs and mins: optimal strategies and best responses exists (compacity and continuity arguments)
2. Against a positional strategy σ, MIN might as well respond positional:

$$\min_{\tau \text{ general}} v_{\sigma,\tau}(x) = \min_{\tau \text{ positional}} v_{\sigma,\tau}(x)$$

3. $\max_{\text{pos}} \min_{\text{pos}} = \max_{\text{pos}} \min_{\text{gen}} \leq \max_{\text{gen}} \min_{\text{gen}} \leq \min_{\text{gen}} \max_{\text{gen}} \leq \min_{\text{pos}} \max_{\text{gen}} = \min_{\text{pos}} \max_{\text{pos}}$

Main lines of a proof ...

1. Sups and infs are maxs and mins: optimal strategies and best responses exists (compacity and continuity arguments)
2. Against a positional strategy σ, MIN might as well respond positional:

$$\min_{\tau \text{ general}} v_{\sigma,\tau}(x) = \min_{\tau \text{ positional}} v_{\sigma,\tau}(x)$$

3. $\max_{\text{pos}} \min_{\text{pos}} = \max_{\text{pos}} \min_{\text{gen}} \leq \max_{\text{gen}} \min_{\text{gen}} \leq \min_{\text{gen}} \max_{\text{gen}} \leq \min_{\text{pos}} \max_{\text{gen}} = \min_{\text{pos}} \max_{\text{pos}}$

Main lines of a proof ...

1. Sups and infs are maxs and mins: optimal strategies and best responses exists (compacity and continuity arguments)
2. Against a positional strategy σ, MIN might as well respond positional:

$$\min_{\tau \text{ general}} v_{\sigma,\tau}(x) = \min_{\tau \text{ positional}} v_{\sigma,\tau}(x)$$

3. $\max_{\text{pos}} \min_{\text{pos}} = \max_{\text{pos}} \min_{\text{gen}} \leq \max_{\text{gen}} \min_{\text{gen}} \leq \min_{\text{gen}} \max_{\text{gen}} \leq \min_{\text{pos}} \max_{\text{gen}} = \min_{\text{pos}} \max_{\text{pos}}$

Main lines of a proof ...

1. Sups and infs are maxs and mins: optimal strategies and best responses exists (compacity and continuity arguments)
2. Against a positional strategy σ, MIN might as well respond positional:

$$\min_{\tau \text{ general}} v_{\sigma,\tau}(x) = \min_{\tau \text{ positional}} v_{\sigma,\tau}(x)$$

3. $\max_{\text{pos}} \min_{\text{pos}} = \max_{\text{pos}} \min_{\text{gen}} \leq \max_{\text{gen}} \min_{\text{gen}} \leq$
   $\min_{\text{gen}} \max_{\text{gen}} \leq \min_{\text{pos}} \max_{\text{gen}} = \min_{\text{pos}} \max_{\text{pos}}$

Main lines of a proof ...

1. Sups and infs are maxs and mins: optimal strategies and best responses exists (compacity and continuity arguments)
2. Against a positional strategy σ, MIN might as well respond positional:

$$\min_{\tau \text{ general}} v_{\sigma,\tau}(x) = \min_{\tau \text{ positional}} v_{\sigma,\tau}(x)$$

3. $\max_{\mathsf{pos}} \min_{\mathsf{pos}} = \max_{\mathsf{pos}} \min_{\mathsf{gen}} \leq \max_{\mathsf{gen}} \min_{\mathsf{gen}} \leq \min_{\mathsf{gen}} \max_{\mathsf{gen}} \leq \min_{\mathsf{pos}} \max_{\mathsf{gen}} = \min_{\mathsf{pos}} \max_{\mathsf{pos}}$

Main lines of a proof ...

1. Sups and infs are maxs and mins: optimal strategies and best responses exists (compacity and continuity arguments)
2. Against a positional strategy σ, MIN might as well respond positional:

$$\min_{\tau \text{ general}} v_{\sigma,\tau}(x) = \min_{\tau \text{ positional}} v_{\sigma,\tau}(x)$$

3. $\max_{\mathsf{pos}} \min_{\mathsf{pos}} = \max_{\mathsf{pos}} \min_{\mathsf{gen}} \leq \max_{\mathsf{gen}} \min_{\mathsf{gen}} \leq \min_{\mathsf{gen}} \max_{\mathsf{gen}} \leq \min_{\mathsf{pos}} \max_{\mathsf{gen}} = \min_{\mathsf{pos}} \max_{\mathsf{pos}}$

Main lines of a proof ...

1. Sups and infs are maxs and mins: optimal strategies and best responses exists (compacity and continuity arguments)
2. Against a positional strategy σ, MIN might as well respond positional:

$$\min_{\tau \text{ general}} v_{\sigma,\tau}(x) = \min_{\tau \text{ positional}} v_{\sigma,\tau}(x)$$

3. $\max_{\text{pos}} \min_{\text{pos}} = \max_{\text{pos}} \min_{\text{gen}} \leq \max_{\text{gen}} \min_{\text{gen}} \leq \min_{\text{gen}} \max_{\text{gen}} \leq \min_{\text{pos}} \max_{\text{gen}} = \min_{\text{pos}} \max_{\text{pos}}$
4. Finite number of strategies $\rightarrow$ zero-sum matrix game

$$\max_{\text{pos}} \min_{\text{pos}} = \min_{\text{pos}} \max_{\text{pos}}$$

Main lines of a proof ...

1. Sups and infs are maxs and mins: optimal strategies and best responses exists (compacity and continuity arguments)
2. Against a positional strategy σ, MIN might as well respond positional:

$$\min_{\tau \text{ general}} v_{\sigma,\tau}(x) = \min_{\tau \text{ positional}} v_{\sigma,\tau}(x)$$

3. $\max_{\mathsf{pos}} \min_{\mathsf{pos}} = \max_{\mathsf{pos}} \min_{\mathsf{gen}} \leq \max_{\mathsf{gen}} \min_{\mathsf{gen}} \leq \min_{\mathsf{gen}} \max_{\mathsf{gen}} \leq \min_{\mathsf{pos}} \max_{\mathsf{gen}} = \min_{\mathsf{pos}} \max_{\mathsf{pos}}$
4. Finite number of strategies $\rightarrow$ zero-sum matrix game

$$\max_{\mathsf{pos}} \min_{\mathsf{pos}} = \min_{\mathsf{pos}} \max_{\mathsf{pos}}$$

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 $v_G(x) > \frac{1}{2}$*

How to stop a game?

Idea of proof:

1. $v_G(x) > \frac{1}{2} \iff v_G(x) \geq \frac{1}{2} + 4^{-n}$
2. values are stable under perturbations,

How to stop a game?

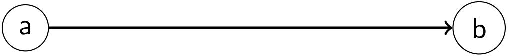
Idea of proof:

1. $v_G(x) > \frac{1}{2} \iff v_G(x) \geq \frac{1}{2} + 4^{-n}$
2. values are stable under perturbations,
3. replace all arcs

How to stop a game?

Idea of proof:

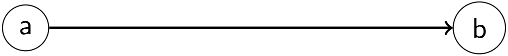
1. $v_G(x) > \frac{1}{2} \iff v_G(x) \geq \frac{1}{2} + 4^{-n}$
2. values are stable under perturbations,
3. replace all arcs



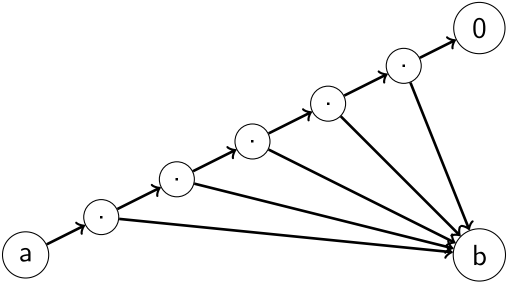
How to stop a game?

Idea of proof:

1. $v_G(x) > \frac{1}{2} \iff v_G(x) \geq \frac{1}{2} + 4^{-n}$
2. values are stable under perturbations,
3. replace all arcs



by



Optimality conditions

A language L is in NP if there is a language $C \in$ P such that

$$x \in L \Leftrightarrow \exists y \in \Sigma^{poly(|x|)}, (x, y) \in C$$

Lemma

G stopping SSG, and σ, τ are optimal strategies if and only if

$$\text{for all } x \in V_{MIN}, \quad v_{\sigma,\tau}(x) = \min(v_{\sigma,\tau}(x_1), v_{\sigma,\tau}(x_2))$$

$$\text{for all } x \in V_{MAX}, \quad v_{\sigma,\tau}(x) = \max(v_{\sigma,\tau}(x_1), v_{\sigma,\tau}(x_2))$$

Optimality conditions

A language L is in NP if there is a language $C \in$ P such that

$$x \in L \Leftrightarrow \exists y \in \Sigma^{poly(|x|)}, (x, y) \in C$$

Lemma

G stopping SSG, and σ, τ are optimal strategies if and only if

$$\text{for all } x \in V_{MIN}, \quad v_{\sigma,\tau}(x) = \min(v_{\sigma,\tau}(x_1), v_{\sigma,\tau}(x_2))$$

$$\text{for all } x \in V_{MAX}, \quad v_{\sigma,\tau}(x) = \max(v_{\sigma,\tau}(x_1), v_{\sigma,\tau}(x_2))$$

Theorem (Condon 89)

The SSG value problem is in NP.

Optimality conditions

A language L is in NP if there is a language $C \in \mathsf{P}$ such that

$$x \in L \Leftrightarrow \exists y \in \Sigma^{poly(|x|)}, (x, y) \in C$$

Lemma

G stopping SSG, and σ, τ are optimal strategies if and only if

$$\textit{for all } x \in V_{MIN}, \quad v_{\sigma,\tau}(x) = \min(v_{\sigma,\tau}(x_1), v_{\sigma,\tau}(x_2))$$

$$\textit{for all } x \in V_{MAX}, \quad v_{\sigma,\tau}(x) = \max(v_{\sigma,\tau}(x_1), v_{\sigma,\tau}(x_2))$$

Theorem (Condon 89)

The SSG value problem is in NP.

Further Complexity Considerations

The symmetry between MAX and MIN put the SSG value problem in coNP.

Another problem in NP $\cap$ coNP: PRIME.

Further Complexity Considerations

The symmetry between MAX and MIN put the SSG value problem in coNP.

Another problem in NP $\cap$ coNP: PRIME.

Lemma

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

Further Complexity Considerations

The symmetry between MAX and MIN put the SSG value problem in coNP.

Another problem in NP $\cap$ coNP: PRIME.

Lemma

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

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

Further Complexity Considerations

The symmetry between MAX and MIN put the SSG value problem in coNP.

Another problem in NP $\cap$ coNP: PRIME.

Lemma

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

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

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

Further Complexity Considerations

The symmetry between MAX and MIN put the SSG value problem in coNP.

Another problem in NP $\cap$ coNP: PRIME.

Lemma

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

The problem is in UP $\cap$ 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 ?

Further Complexity Considerations

The symmetry between MAX and MIN put the SSG value problem in coNP.

Another problem in $\mathsf{NP} \cap \mathsf{coNP}$: PRIME.

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 ?

Introduction to Games

Fundamental Properties of SSGs and Complexity Classes

Algorithms to solve SSG

Average Only

A simple case: a SSG with only average vertices.

It is equivalent to a Markov process and a SSG with a fixed strategy for MIN and MAX.

Average Only

A simple case: a SSG with only average vertices.

It is equivalent to a Markov process and a SSG with a fixed strategy for MIN and MAX.

Values can be represented by a linear system and solved in polynomial time. For each vertex x with outvertices x_1 and x_2,

$$v(x) = \frac{1}{2}v(x_1) + \frac{1}{2}v(x_2)$$

where the values of sinks are replaced by 0 or 1.

Average Only

A simple case: a SSG with only average vertices.

It is equivalent to a Markov process and a SSG with a fixed strategy for MIN and MAX.

Values can be represented by a linear system and solved in polynomial time. For each vertex x with outvertices x_1 and x_2,

$$v(x) = \frac{1}{2}v(x_1) + \frac{1}{2}v(x_2)$$

where the values of sinks are replaced by 0 or 1.

We use that to compute $v_{\sigma,\tau}(x)$.

Average Only

A simple case: a SSG with only average vertices.

It is equivalent to a Markov process and a SSG with a fixed strategy for MIN and MAX.

Values can be represented by a linear system and solved in polynomial time. For each vertex x with outvertices x_1 and x_2,

$$v(x) = \frac{1}{2}v(x_1) + \frac{1}{2}v(x_2)$$

where the values of sinks are replaced by 0 or 1.

We use that to compute $v_{\sigma,\tau}(x)$.

Average and MIN

We consider SSG with MIN and average vertices only.
Equivalent to a SSG with a fixed strategy for MAX.

The switch:
Let x be a MIN vertex. Suppose
$v_\tau(x) = v_\tau(x_1) > v_\tau(x_2))$

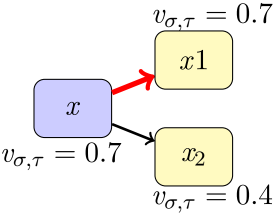
Average and MIN

We consider SSG with MIN and average vertices only.
Equivalent to a SSG with a fixed strategy for MAX.

The switch:

Let x be a MIN vertex. Suppose
$v_\tau(x) = v_\tau(x_1) > v_\tau(x_2))$

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



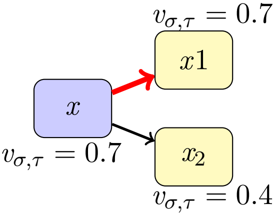
Average and MIN

We consider SSG with MIN and average vertices only.
Equivalent to a SSG with a fixed strategy for MAX.

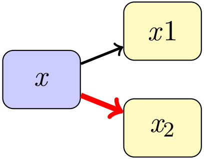
The switch:

Let x be a MIN vertex. Suppose
$v_\tau(x) = v_\tau(x_1) > v_\tau(x_2))$

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



A switch is profitable for MIN: $\tau' < \tau$

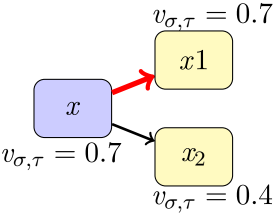


Average and MIN

We consider SSG with MIN and average vertices only.
Equivalent to a SSG with a fixed strategy for MAX.

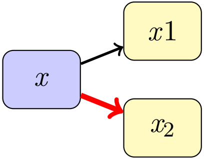
The switch:

Let x be a MIN vertex. Suppose
$v_\tau(x) = v_\tau(x_1) > v_\tau(x_2))$



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

A switch is profitable for MIN: $\tau' < \tau$

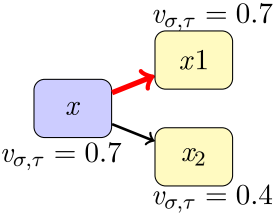


Algorithm to find an optimal strategy: keep switching.

Average and MIN

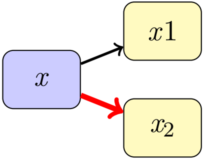
We consider SSG with MIN and average vertices only.
Equivalent to a SSG with a fixed strategy for MAX.

<u>The switch:</u>
Let x be a MIN vertex. Suppose
$v_\tau(x) = v_\tau(x_1) > v_\tau(x_2))$



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

A switch is profitable for MIN: $\tau' < \tau$



Algorithm to find an optimal strategy: keep switching.

Faster Computation of a Best Response

$$F_\sigma : \begin{cases} [0,1]^V & \longrightarrow & [0,1]^V \\ v_x & \longmapsto & \begin{cases} \min(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MIN} \\ v_{\sigma(x)} \text{ if } x \in V_{MAX} \\ \frac{1}{2} v_{x_1} + \frac{1}{2} v_{x_2} \text{ if } x \in V_{AVE} \end{cases} \end{cases}$$

where the values of sinks are replaced by 0 or 1.

- Operator F_σ is contracting (sup norm)
  $\rightarrow$ single fixed point = value vector of σ

Faster Computation of a Best Response

$$F_\sigma : \begin{cases} [0,1]^V & \longrightarrow & [0,1]^V \\ v_x & \longmapsto & \begin{cases} \min(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MIN} \\ v_{\sigma(x)} \text{ if } x \in V_{MAX} \\ \frac{1}{2} v_{x_1} + \frac{1}{2} v_{x_2} \text{ if } x \in V_{AVE} \end{cases} \end{cases}$$

where the values of sinks are replaced by 0 or 1.

- Operator F_σ is contracting (sup norm)
  $\rightarrow$ single fixed point = value vector of σ
- Solving $F_\sigma v = v$ by linear programing

$$\max \sum_i v_i$$

$$F_\sigma(v) \leq v$$

Faster Computation of a Best Response

$$F_\sigma : \begin{cases} [0,1]^V & \longrightarrow & [0,1]^V \\ v_x & \longmapsto & \begin{cases} \min(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MIN} \\ v_{\sigma(x)} \text{ if } x \in V_{MAX} \\ \frac{1}{2} v_{x_1} + \frac{1}{2} v_{x_2} \text{ if } x \in V_{AVE} \end{cases} \end{cases}$$

where the values of sinks are replaced by 0 or 1.

- Operator F_σ is contracting (sup norm)
  $\rightarrow$ single fixed point $=$ value vector of σ
- Solving $F_\sigma v = v$ by linear programing

$$\max \sum_i v_i$$

$$F_\sigma(v) \leq v$$

Polytime algorithm to compute $v_\sigma(x)$.

Faster Computation of a Best Response

$$F_\sigma : \begin{cases} [0,1]^V & \longrightarrow & [0,1]^V \\ v_x & \longmapsto & \begin{cases} \min(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MIN} \\ v_{\sigma(x)} \text{ if } x \in V_{MAX} \\ \frac{1}{2} v_{x_1} + \frac{1}{2} v_{x_2} \text{ if } x \in V_{AVE} \end{cases} \end{cases}$$

where the values of sinks are replaced by 0 or 1.

- Operator F_σ is contracting (sup norm)
  $\rightarrow$ single fixed point $=$ value vector of σ
- Solving $F_\sigma v = v$ by linear programing

$$\max \sum_i v_i$$

$$F_\sigma(v) \leq v$$

Polytime algorithm to compute $v_\sigma(x)$.

Fixpoint

A generalization of the fixpoint method to SSG:

$$F : \begin{cases} [0,1]^V & \longrightarrow & [0,1]^V \\ v_x & \longmapsto & \begin{cases} \min(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MIN} \\ \max(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MAX} \\ \frac{1}{2} v_{x_1} + \frac{1}{2} v_{x_2} \text{ if } x \in V_{AVE} \end{cases} \end{cases}$$

where the values of sinks are replaced by 0 or 1.

The operator F is contracting and its fixpoint is the optimal value vector.

Fixpoint

A generalization of the fixpoint method to SSG:

$$F : \begin{cases} [0,1]^V & \longrightarrow & [0,1]^V \\ v_x & \longmapsto & \begin{cases} \min(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MIN} \\ \max(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MAX} \\ \frac{1}{2} v_{x_1} + \frac{1}{2} v_{x_2} \text{ if } x \in V_{AVE} \end{cases} \end{cases}$$

where the values of sinks are replaced by 0 or 1.

The operator F is contracting and its fixpoint is the optimal value vector.

Computing the values is in the class PPAD.
Does not converge fast.

Fixpoint

A generalization of the fixpoint method to SSG:

$$F : \begin{cases} [0,1]^V & \longrightarrow & [0,1]^V \\ v_x & \longmapsto & \begin{cases} \min(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MIN} \\ \max(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MAX} \\ \frac{1}{2} v_{x_1} + \frac{1}{2} v_{x_2} \text{ if } x \in V_{AVE} \end{cases} \end{cases}$$

where the values of sinks are replaced by 0 or 1.

The operator F is contracting and its fixpoint is the optimal value vector.

Computing the values is in the class PPAD.
Does not converge fast.

Fixpoint

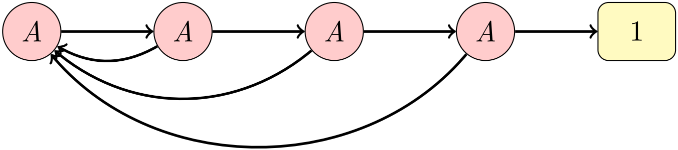
A generalization of the fixpoint method to SSG:

$$F : \begin{cases} [0,1]^V & \longrightarrow \quad [0,1]^V \\ v_x & \longmapsto \quad \begin{cases} \min(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MIN} \\ \max(v_{x_1}, v_{x_2}) \text{ if } x \in V_{MAX} \\ \frac{1}{2} v_{x_1} + \frac{1}{2} v_{x_2} \text{ if } x \in V_{AVE} \end{cases} \end{cases}$$

where the values of sinks are replaced by 0 or 1.

The operator F is contracting and its fixpoint is the optimal value vector.

Computing the values is in the class PPAD.
Does not converge fast.



Hoffman-Karp Algorithm

The strategy improvement algorithm or Hoffman-Karp algorithm:

1. choose σ_0 and let $\tau_0 = \tau(\sigma_0)$ (best response)
2. while (σ_k, τ_k) is not optimal, obtain σ_{k+1} by switching σ_k ; let $\tau_{k+1} = \tau(\sigma_{k+1})$

Lemma

For all k, $v_{\sigma_{k+1},\tau_{k+1}} > v_{\sigma_k,\tau_k}$

Hoffman-Karp Algorithm

The strategy improvement algorithm or Hoffman-Karp algorithm:

1. choose σ_0 and let $\tau_0 = \tau(\sigma_0)$ (best response)
2. while (σ_k, τ_k) is not optimal, obtain σ_{k+1} by switching σ_k ; let $\tau_{k+1} = \tau(\sigma_{k+1})$

Lemma

For all k, $v_{\sigma_{k+1},\tau_{k+1}} > v_{\sigma_k,\tau_k}$

Theorem (Tripathi, Valkanova, Kumar)

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

Computing the value is thus in PLS but the algorithm can take exponential time:

- Friedmann (2009) gives a counter-example for parity game with $2^{\sqrt{n}}$ iterations, claimed 2^{cn}.
- Andersson (2009) shows that this counterexample survives the reduction

Hoffman-Karp Algorithm

The strategy improvement algorithm or Hoffman-Karp algorithm:

1. choose σ_0 and let $\tau_0 = \tau(\sigma_0)$ (best response)
2. while (σ_k, τ_k) is not optimal, obtain σ_{k+1} by switching σ_k ; let $\tau_{k+1} = \tau(\sigma_{k+1})$

Lemma

For all k, $v_{\sigma_{k+1},\tau_{k+1}} > v_{\sigma_k,\tau_k}$

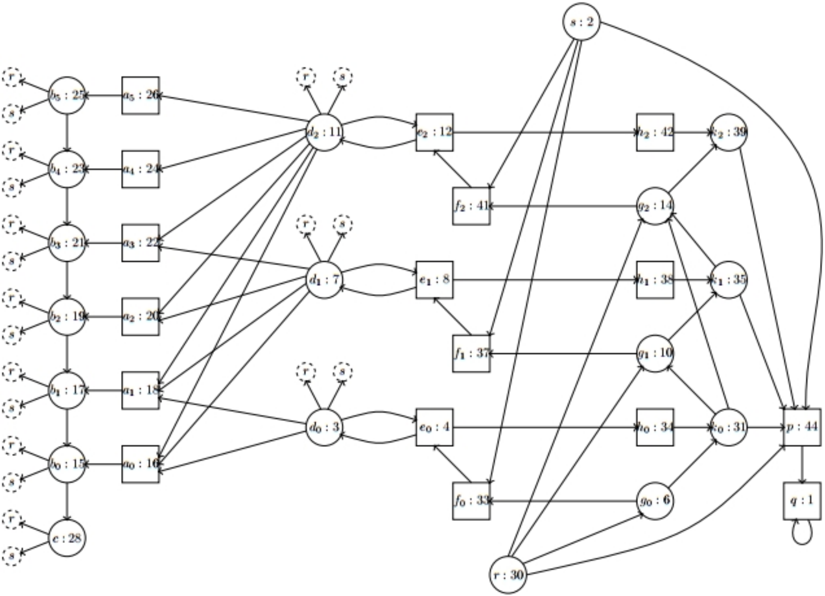
Theorem (Tripathi, Valkanova, Kumar)

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

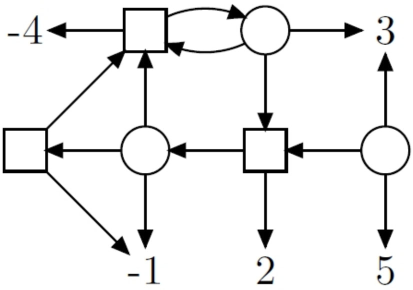
Computing the value is thus in PLS but the algorithm can take exponential time:

- Friedmann (2009) gives a counter-example for parity game with $2^{\sqrt{n}}$ iterations, claimed 2^{cn}.
- Andersson (2009) shows that this counterexample survives the reduction

Counter-Example



No average vertices



Deterministic graphical games (Washburn 1966, Andersson et al. 2012)

Definition = SSG without average vertices, but allow sinks with arbitrary payoffs

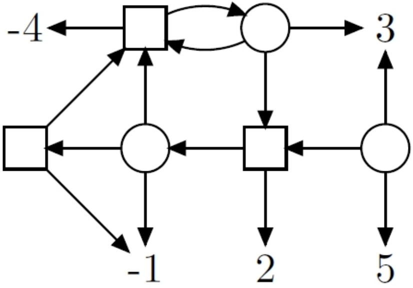
Solving DGG in linear time by backtracking

While possible :

1. sink s with maximal payoff: incoming MIN arcs never go there if they have a choice: delete arc or merge
2. Do the opposite for the minimum payoff sink.

In the end, the vertices with no connection to sinks have value 0.

No average vertices



Deterministic graphical games (Washburn 1966, Andersson et al. 2012)

Definition = SSG without average vertices, but allow sinks with arbitrary payoffs

Solving DGG in linear time by backtracking

While possible :

1. sink s with maximal payoff: incoming MIN arcs never go there if they have a choice: delete arc or merge
2. Do the opposite for the minimum payoff sink.

In the end, the vertices with no connection to sinks have value 0.

Few average vertices

Theorem (Gimbert and Horn 2009)

There is an algorithm which computes values and optimal strategies of SSGs with n vertices and k average vertices in time $O(k!n)$.

(Moreover the outdegree of nodes is unlimited)

- A strategy consists in choosing among nodes. Hence a preference order on all nodes yields a strategy.

Few average vertices

Theorem (Gimbert and Horn 2009)

There is an algorithm which computes values and optimal strategies of SSGs with n vertices and k average vertices in time $O(k!n)$.

(Moreover the outdegree of nodes is unlimited)

- A strategy consists in choosing among nodes. Hence a preference order on all nodes yields a strategy.
- An order on V_{AVE} is enough.

$$0 < a_1 < a_2 \cdots a_k < 1$$

MAX tries to force the next average vertex to be large.
MIN tries to force the next average vertex to be small.

Few average vertices

Theorem (Gimbert and Horn 2009)

There is an algorithm which computes values and optimal strategies of SSGs with n vertices and k average vertices in time $O(k!n)$.

(Moreover the outdegree of nodes is unlimited)

- A strategy consists in choosing among nodes. Hence a preference order on all nodes yields a strategy.
- An order on V_{AVE} is enough.

$$0 < a_1 < a_2 \cdots a_k < 1$$

MAX tries to force the next average vertex to be large.
MIN tries to force the next average vertex to be small.

Directed Acyclic Graphs

A directed acyclic graph is a graph without a directed cycle.

Algorithm:

The sinks are initialized to 0 and 1

While possible:

- $x \in V_{MAX}$, $v(x) = \max(v(x_1), v(x_2))$
- $x \in V_{MIN}$, $v(x) = \min(v(x_1), v(x_2))$
- $x \in V_{AVE}$, $v(x) = \frac{1}{2}v(x_1) + \frac{1}{2}v(x_2)$

Directed Acyclic Graphs

A directed acyclic graph is a graph without a directed cycle.

Algorithm:

The sinks are initialized to 0 and 1

While possible:

- $x \in V_{MAX}$, $v(x) = \max(v(x_1), v(x_2))$
- $x \in V_{MIN}$, $v(x) = \min(v(x_1), v(x_2))$
- $x \in V_{AVE}$, $v(x) = \frac{1}{2}v(x_1) + \frac{1}{2}v(x_2)$

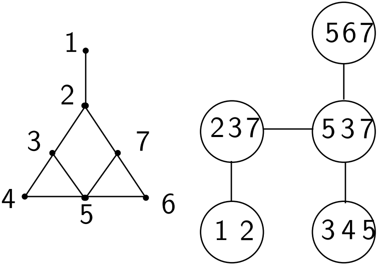
Almost Acyclic: Tree-Width

Definition (Tree Decomposition)

A tree decomposition of a graph G is a pair (T, X) where $X = \{X_1, \ldots, X_n\}$ is a family of subsets (or bags) of $V(G)$ and T is a tree whose nodes are the X_i such that:

- the union of the X_i equals $V(G)$
- every edge $(u, v) \in E(G)$ is included in some X_i.
- for each u in $V(G)$ the set of X_i which contains u is connex.

width $= \max_i |X_i|$

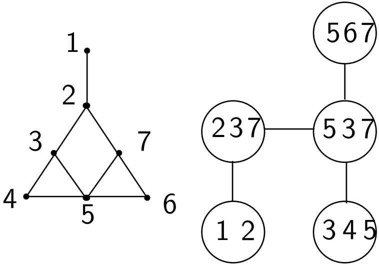


Almost Acyclic: Tree-Width

Definition (Tree Decomposition)

A tree decomposition of a graph G is a pair (T, X) where $X = \{X_1, \ldots, X_n\}$ is a family of subsets (or bags) of $V(G)$ and T is a tree whose nodes are the X_i such that:

- the union of the X_i equals $V(G)$
- every edge $(u, v) \in E(G)$ is included in some X_i.
- for each u in $V(G)$ the set of X_i which contains u is connex.



width $= \max_i |X_i|$

treewidth $= \min_{(T,X)} width(T, X)$

Almost Acyclic: Tree-Width

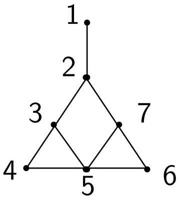
Definition (Tree Decomposition)

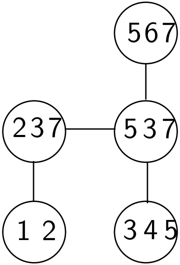
A tree decomposition of a graph G is a pair (T, X) where $X = \{X_1, \ldots, X_n\}$ is a family of subsets (or bags) of $V(G)$ and T is a tree whose nodes are the X_i such that:

- the union of the X_i equals $V(G)$
- every edge $(u, v) \in E(G)$ is included in some X_i.
- for each u in $V(G)$ the set of X_i which contains u is connex.

width $= \max_i |X_i|$

treewidth $= \min_{(T,X)} width(T, X)$





Solving Bounded Treewidth SSGs

Theorem (Work in progress)

For all $k \in \mathbb{N}$, the SSG value problem is in P *when restricted to SSGs of treewidth bounded by k.*

The complexity of the algorithm is in $O(k2^{k^2}n)$.

- Notion of directed treewidth to capture DAG and adaptation to the SSG case.

Solving Bounded Treewidth SSGs

Theorem (Work in progress)

For all $k \in \mathbb{N}$, the SSG value problem is in P *when restricted to SSGs of treewidth bounded by k.*

The complexity of the algorithm is in $O(k2^{k^2}n)$.

- Notion of directed treewidth to capture DAG and adaptation to the SSG case.
- Improve the algorithm to be less dependent of k.

Solving Bounded Treewidth SSGs

Theorem (Work in progress)

For all $k \in \mathbb{N}$, the SSG value problem is in P *when restricted to SSGs of treewidth bounded by k.*

The complexity of the algorithm is in $O(k2^{k^2}n)$.

- Notion of directed treewidth to capture DAG and adaptation to the SSG case.
- Improve the algorithm to be less dependent of k.
- Use ideas to get another way to solve SSG with few average vertices.

Solving Bounded Treewidth SSGs

Theorem (Work in progress)

For all $k \in \mathbb{N}$, the SSG value problem is in P *when restricted to SSGs of treewidth bounded by k.*

The complexity of the algorithm is in $O(k2^{k^2}n)$.

- Notion of directed treewidth to capture DAG and adaptation to the SSG case.
- Improve the algorithm to be less dependent of k.
- Use ideas to get another way to solve SSG with few average vertices.
- Is the SSG value problem expressible in MSO over graph?

Solving Bounded Treewidth SSGs

Theorem (Work in progress)

For all $k \in \mathbb{N}$, the SSG value problem is in P *when restricted to SSGs of treewidth bounded by k.*

The complexity of the algorithm is in $O(k2^{k^2}n)$.

- Notion of directed treewidth to capture DAG and adaptation to the SSG case.
- Improve the algorithm to be less dependent of k.
- Use ideas to get another way to solve SSG with few average vertices.
- Is the SSG value problem expressible in MSO over graph?

Thanks.