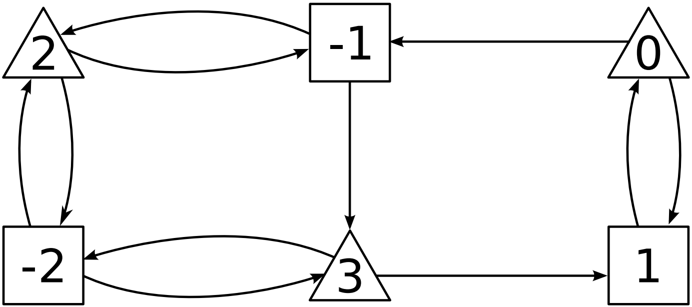
Non-oblivious Strategy Improvement

John Fearnley

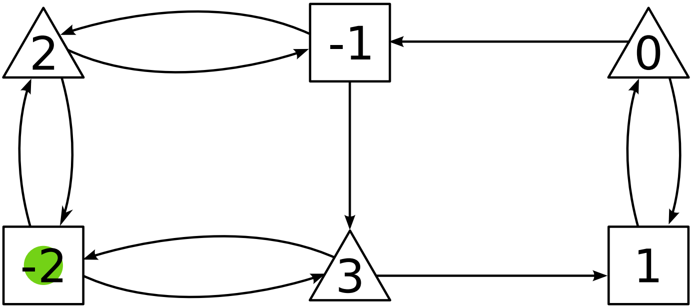
University of Warwick

GASICS Meeting 23rd October 2009

Mean-Payoff Games

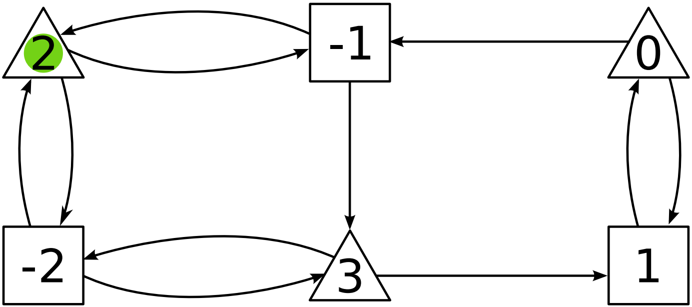


Mean-Payoff Games



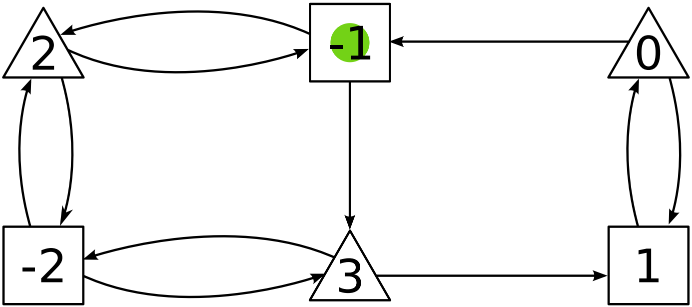
Rewards = -2,

Mean-Payoff Games



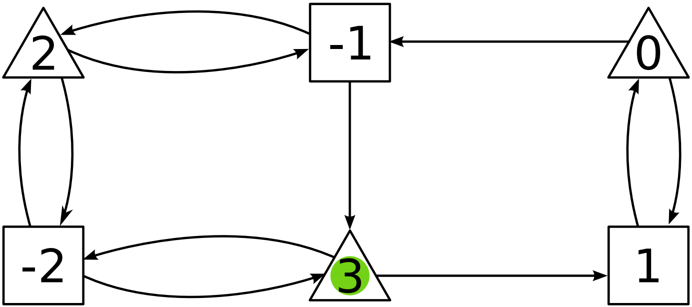
Rewards = -2, 2,

Mean-Payoff Games



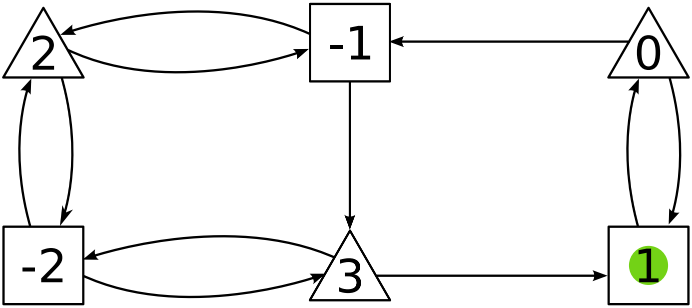
Rewards = -2, 2, -1,

Mean-Payoff Games



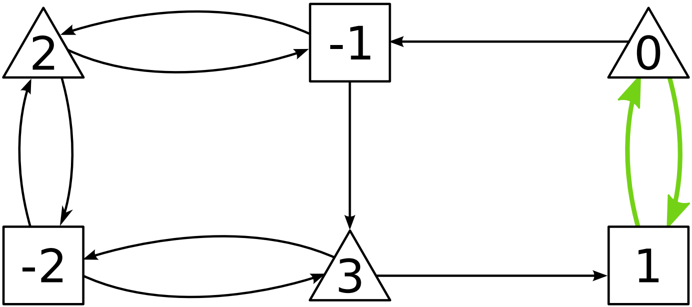
Rewards = -2, 2, -1, 3,

Mean-Payoff Games



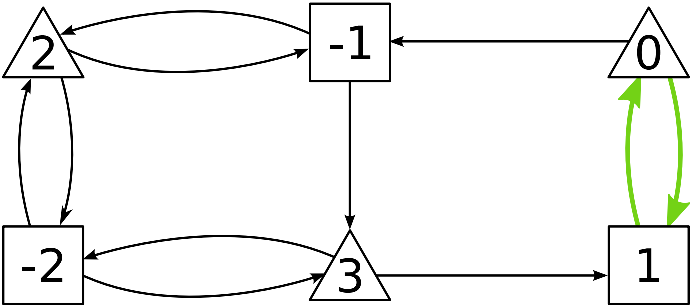
Rewards = -2, 2, -1, 3, 1,

Mean-Payoff Games



Rewards = -2, 2, -1, 3, 1, 0, 1, 0, 1 . . .

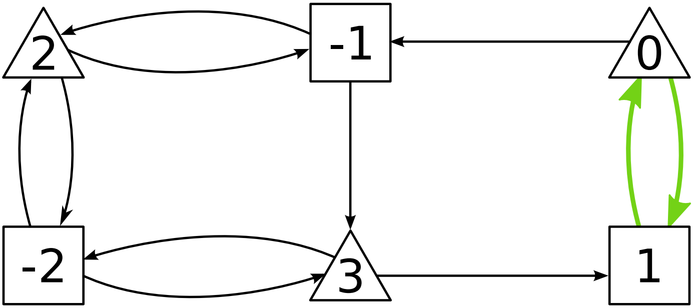
Mean-Payoff Games



Rewards = -2, 2, -1, 3, 1, 0, 1, 0, 1 ...

Payoff $= \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} r_i$

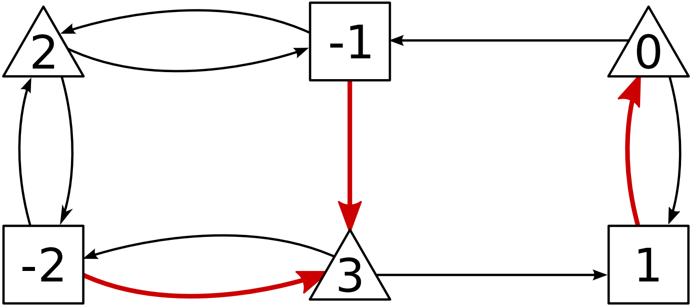
Mean-Payoff Games



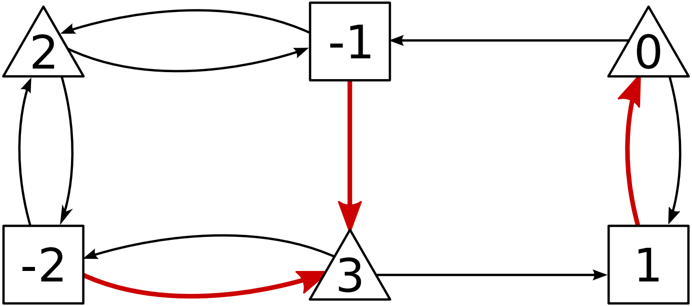
Rewards = -2, 2, -1, 3, 1, 0, 1, 0, 1 . . .

Payoff $= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} r_i = 0.5$

Positional Strategies



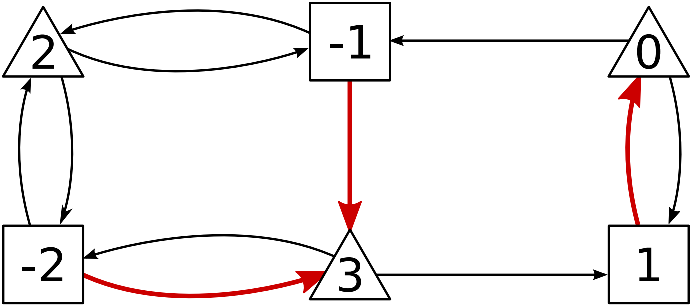
Positional Strategies



Zero Mean Partition Problem

For each vertex, does Max have a positional strategy that guarantees payoff > 0?

Positional Strategies



Zero Mean Partition Problem

For each vertex, does Max have a positional strategy that guarantees payoff > 0?

$\Leftrightarrow$

Does Max have a strategy against which every cycle is positive?

Motivation

An algorithm for zero mean partition can be used to find the exact value of each vertex. (Björklund and Vorobyov 2007)

Motivation

An algorithm for zero mean partition can be used to find the exact value of each vertex. (Björklund and Vorobyov 2007)

- Mean payoff games have applications in online scheduling and string comparison problems.

Motivation

An algorithm for zero mean partition can be used to find the exact value of each vertex. (Björklund and Vorobyov 2007)

- Mean payoff games have applications in online scheduling and string comparison problems.
- Parity games can be reduced to mean payoff games.
- Model checking μ-calculus is polynomial time equivalent to the problem of solving a parity game.

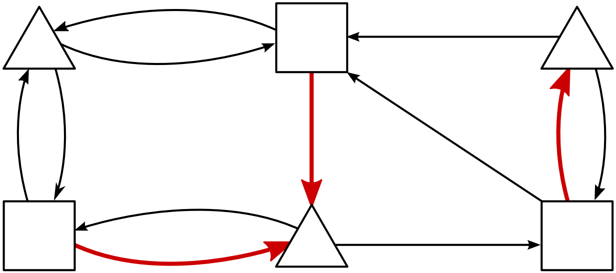
Motivation

An algorithm for zero mean partition can be used to find the exact value of each vertex. (Björklund and Vorobyov 2007)

- Mean payoff games have applications in online scheduling and string comparison problems.
- Parity games can be reduced to mean payoff games.
- Model checking μ-calculus is polynomial time equivalent to the problem of solving a parity game.
- The problem is in NP $\cap$ co-NP but no polynomial time algorithm is known.

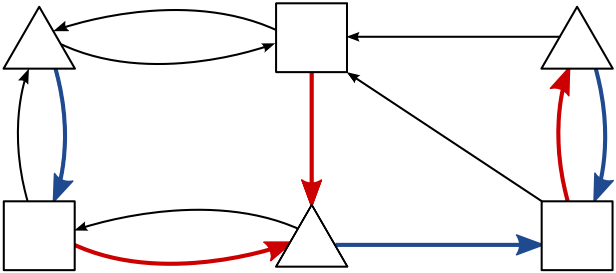
Strategy Improvement

Strategy Improvement



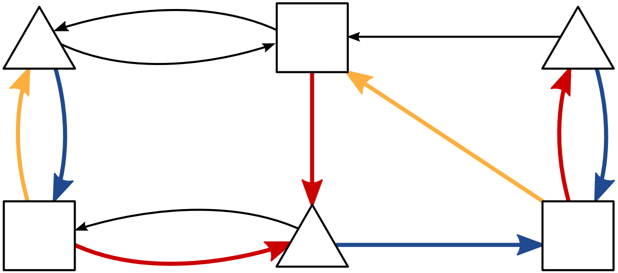
- Pick a strategy for one of the players.

Strategy Improvement



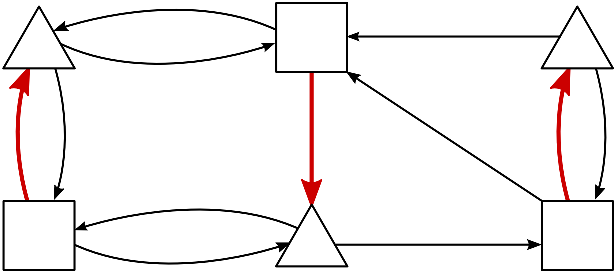
- Pick a strategy for one of the players.
- Compute a best response for the other player.

Strategy Improvement



- Pick a strategy for one of the players.
- Compute a best response for the other player.
- Find a set of profitable edges.

Strategy Improvement



- Pick a strategy for one of the players.
- Compute a best response for the other player.
- Find a set of profitable edges.
- Switch some profitable edges to create an improved strategy.

Strategy Improvement

There are strategy improvement algorithms for:

- Simple stochastic games (Condon 1993)
- Discounted games (Puri 1995)
- Parity games (Vöge and Jurdziński 2000)
- Zero mean partition (Björklund and Vorobyov 2007)

Strategy Improvement

There are strategy improvement algorithms for:

- Simple stochastic games (Condon 1993)
- Discounted games (Puri 1995)
- Parity games (Vöge and Jurdziński 2000)
- Zero mean partition (Björklund and Vorobyov 2007)

Zero Mean Partition

We use Max as the strategy improver.

Zero Mean Partition

We use Max as the strategy improver.

There are two types of strategies:

- If every cycle that Min can form is positive then the strategy is winning.

Zero Mean Partition

We use Max as the strategy improver.

There are two types of strategies:

- If every cycle that Min can form is positive then the strategy is winning.
- Otherwise, Min can form a negative cycle, and the strategy must be improved.

Zero Mean Partition

We use Max as the strategy improver.

There are two types of strategies:

- If every cycle that Min can form is positive then the strategy is winning.
- Otherwise, Min can form a negative cycle, and the strategy must be improved.

We allow Max to break a negative cycle by moving to a sink vertex.

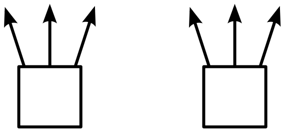
Zero Mean Partition

We use Max as the strategy improver.

There are two types of strategies:

- If every cycle that Min can form is positive then the strategy is winning.
- Otherwise, Min can form a negative cycle, and the strategy must be improved.

We allow Max to break a negative cycle by moving to a sink vertex.



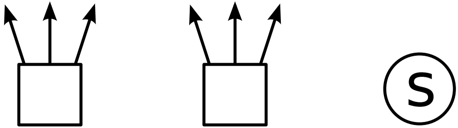
Zero Mean Partition

We use Max as the strategy improver.

There are two types of strategies:

- If every cycle that Min can form is positive then the strategy is winning.
- Otherwise, Min can form a negative cycle, and the strategy must be improved.

We allow Max to break a negative cycle by moving to a sink vertex.



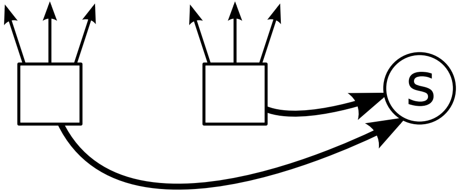
Zero Mean Partition

We use Max as the strategy improver.

There are two types of strategies:

- If every cycle that Min can form is positive then the strategy is winning.
- Otherwise, Min can form a negative cycle, and the strategy must be improved.

We allow Max to break a negative cycle by moving to a sink vertex.

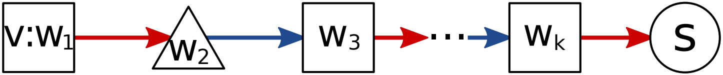


Valuations

When Max plays σ and Min plays τ there are two possibilities.

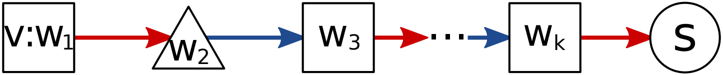
Valuations

When Max plays σ and Min plays τ there are two possibilities.



Valuations

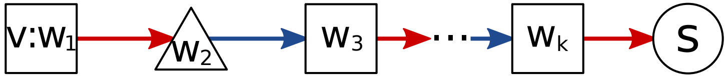
When Max plays σ and Min plays τ there are two possibilities.



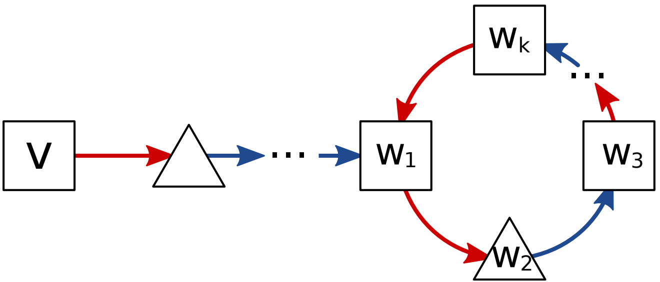
$\text{Val}^{\sigma,\tau}(v) = \sum_{i=1}^{k} w_i$

Valuations

When Max plays σ and Min plays τ there are two possibilities.

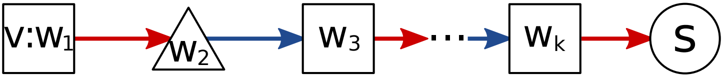


$\mathrm{Val}^{\sigma,\tau}(v) = \sum_{i=1}^{k} w_i$

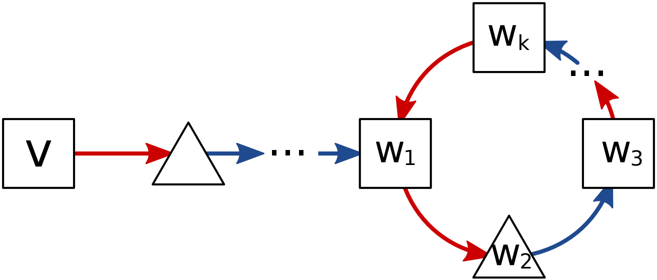


Valuations

When Max plays σ and Min plays τ there are two possibilities.



$\text{Val}^{\sigma,\tau}(v) = \sum_{i=1}^{k} w_i$



$$\text{Val}^{\sigma,\tau}(v) = \begin{cases} \infty & \text{if } \sum_{i=1}^{k} w_i > 0 \\ -\infty & \text{otherwise} \end{cases}$$

Best Response

The best response to a Max strategy is an optimal counter strategy for Min.

Best Response

The best response to a Max strategy is an optimal counter strategy for Min.

For a fixed Max strategy σ, the best response $\mathsf{br}(\sigma)$ satisfies:

$$\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v) \leq \mathsf{Val}^{\sigma,\tau}(v)$$

for all v and τ.

Best Response

The best response to a Max strategy is an optimal counter strategy for Min.

For a fixed Max strategy σ, the best response $\mathsf{br}(\sigma)$ satisfies:

$$\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v) \leq \mathsf{Val}^{\sigma,\tau}(v)$$

for all v and τ.

Proposition (Björklund and Vorobyov 2007)

For every strategy σ the best response $\mathsf{br}(\sigma)$ can be computed in polynomial time.

Valuations

Strategy improvement always considers $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v)$.

Valuations

Strategy improvement always considers $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v)$.

- If $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v) = \infty$ then we have solved the zero mean partition problem for v.

Valuations

Strategy improvement always considers $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v)$.

- If $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v) = \infty$ then we have solved the zero mean partition problem for v.
- Otherwise, the strategy is not good enough, and so it must be improved.

Improving Strategies

Strategies will be improved by switching profitable edges.

Improving Strategies

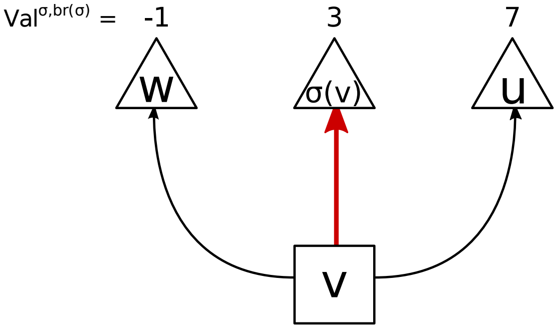
Strategies will be improved by switching profitable edges.

An edge (v, u) is profitable if $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(u) > \mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(\sigma(v))$

Improving Strategies

Strategies will be improved by switching profitable edges.

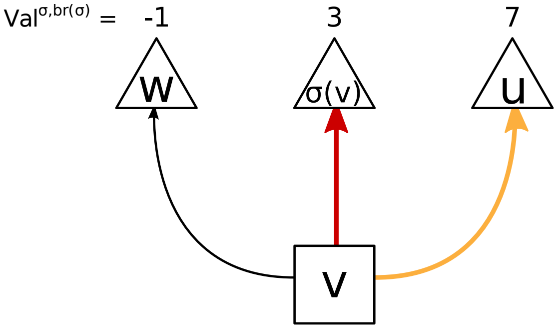
An edge (v, u) is profitable if $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(u) > \mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(\sigma(v))$



Improving Strategies

Strategies will be improved by switching profitable edges.

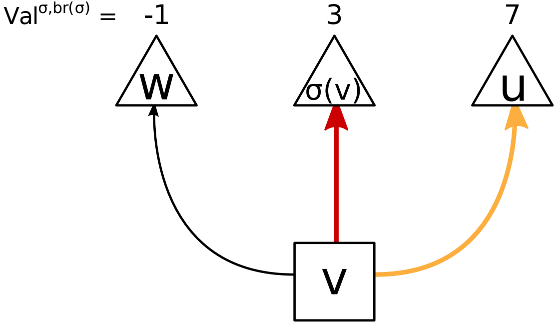
An edge (v, u) is profitable if $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(u) > \mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(\sigma(v))$



Improving Strategies

Strategies will be improved by switching profitable edges.

An edge (v, u) is profitable if $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(u) > \mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(\sigma(v))$

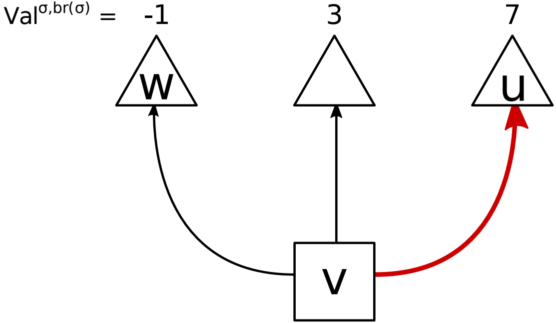


We improve a strategy by using a profitable edge instead of the current successor.

Improving Strategies

Strategies will be improved by switching profitable edges.

An edge (v, u) is profitable if $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(u) > \mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(\sigma(v))$



We improve a strategy by using a profitable edge instead of the current successor.

Strategy Improvement

Theorem (Björklund and Vorobyov 2007)

Switching any subset of profitable edges gives an improved strategy.

Strategy Improvement

Theorem (Björklund and Vorobyov 2007)

Switching any subset of profitable edges gives an improved strategy.

Let P be some subset of profitable edges in σ, and let σ' be σ with every edge in P switched. We have:

$$\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v) \leq \mathsf{Val}^{\sigma',\mathsf{br}(\sigma')}(v)$$

and for some v the inequality is strict.

Strategy Improvement

Theorem (Björklund and Vorobyov 2007)

Switching any subset of profitable edges gives an improved strategy.

Let P be some subset of profitable edges in σ, and let σ' be σ with every edge in P switched. We have:

$$\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v) \leq \mathsf{Val}^{\sigma',\mathsf{br}(\sigma')}(v)$$

and for some v the inequality is strict.

Theorem (Björklund and Vorobyov 2007)

A strategy with no profitable edges is optimal.

Strategy Improvement

Strategy improvement algorithms look like this:

while σ has a profitable edge **do**
  Compute the best response $\text{br}(\sigma)$.
  Compute $\text{Val}^{\sigma,\text{br}(\sigma)}(v)$ for every vertex v.
  Compute the set of profitable edges in σ.
  $\sigma := \sigma$ with some subset of profitable edges switched.
end while

Strategy Improvement

Strategy improvement algorithms look like this:

while σ has a profitable edge **do**
  Compute the best response $\text{br}(\sigma)$.
  Compute $\text{Val}^{\sigma,\text{br}(\sigma)}(v)$ for every vertex v.
  Compute the set of profitable edges in σ.
  $\sigma := \sigma$ with some subset of profitable edges switched.
end while

Strategy Improvement

Strategy improvement algorithms look like this:

while σ has a profitable edge **do**
  Compute the best response $\mathsf{br}(\sigma)$.
  Compute $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v)$ for every vertex v.
  Compute the set of profitable edges in σ.
  $\sigma := \sigma$ with some subset of profitable edges switched.
end while

A switching policy picks the subset that is switched.

Switching Policies

Single-vertex switching policies.

- Can take exponential time. (Lebedev 1988)
- A randomized single-vertex policy is subexponential. $O(2^{\sqrt{(n \log n)}})$ (Björklund and Vorobyov 2007)

Switching Policies

Single-vertex switching policies.

- Can take exponential time. (Lebedev 1988)
- A randomized single-vertex policy is subexponential. $O(2^{\sqrt{(n \log n)}})$ (Björklund and Vorobyov 2007)

All-vertex switching policies.

- Best upper bound of $O(2^n/n)$. (Mansour and Singh 1999)
- Works very well in practice.

Switching Policies

Single-vertex switching policies.

- Can take exponential time. (Lebedev 1988)
- A randomized single-vertex policy is subexponential. $O(2^{\sqrt{(n \log n)}})$ (Björklund and Vorobyov 2007)

All-vertex switching policies.

- Best upper bound of $O(2^n/n)$. (Mansour and Singh 1999)
- Works very well in practice.
- Recently shown to take exponential time. (Friedmann 2009)

Optimal Switching Policies

An optimal subset of profitable edges is the subset that gives the largest possible increase in valuation.

Optimal Switching Policies

An optimal subset of profitable edges is the subset that gives the largest possible increase in valuation.

Let σ' be σ with an optimal subset of profitable edges switched, and χ be σ with some other subset switched. We have:

$$\mathsf{Val}^{\chi,\mathsf{br}(\chi)}(v) \leq \mathsf{Val}^{\sigma',\mathsf{br}(\sigma')}(v)$$

for every vertex v.

Optimal Switching Policies

An optimal subset of profitable edges is the subset that gives the largest possible increase in valuation.

Let σ' be σ with an optimal subset of profitable edges switched, and χ be σ with some other subset switched. We have:

$$\mathsf{Val}^{\chi,\mathsf{br}(\chi)}(v) \leq \mathsf{Val}^{\sigma',\mathsf{br}(\sigma')}(v)$$

for every vertex v.

Theorem (Schewe 2008)

An optimal subset of edges can be computed in polynomial time.

Optimal Switching Policies

An optimal subset of profitable edges is the subset that gives the largest possible increase in valuation.

Let σ' be σ with an optimal subset of profitable edges switched, and χ be σ with some other subset switched. We have:

$$\mathsf{Val}^{\chi,\mathsf{br}(\chi)}(v) \leq \mathsf{Val}^{\sigma',\mathsf{br}(\sigma')}(v)$$

for every vertex v.

Theorem (Schewe 2008)

An optimal subset of edges can be computed in polynomial time.

Theorem (Friedmann 2009)

Optimal switching policies are exponential.

Strategy Improvement

So far:

- Most reasonable switching policies are super polynomial.
- Even picking the best possible subset doesn't work.

Strategy Improvement

So far:

- Most reasonable switching policies are super polynomial.
- Even picking the best possible subset doesn't work.

Our contribution

We claim that previous policies fail because they are oblivious.

Strategy Improvement

So far:

- Most reasonable switching policies are super polynomial.
- Even picking the best possible subset doesn't work.

Our contribution

We claim that previous policies fail because they are oblivious.

We define non-oblivious strategy improvement.

Non-oblivious strategy improvement

For non-oblivious strategy improvement we need to know:

- what can be remembered.
- how this can be used in later iterations.

Non-oblivious strategy improvement

For non-oblivious strategy improvement we need to know:

- what can be remembered.
- how this can be used in later iterations.

Our answers to these questions come from profitable back edges.

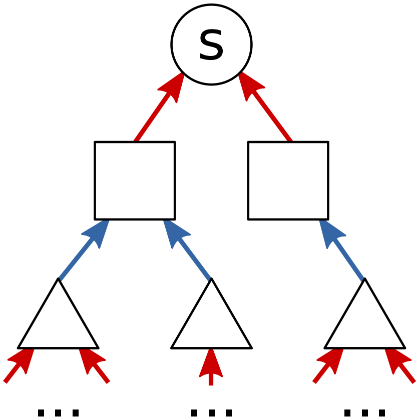
Strategy Trees

The strategy tree is a representation of the current strategy and best response for the vertices with finite valuation.

Strategy Trees

The strategy tree is a representation of the current strategy and best response for the vertices with finite valuation.

The tree of a Max strategy σ is a tree rooted at the sink whose edges are those chosen by σ and $\mathrm{br}(\sigma)$.



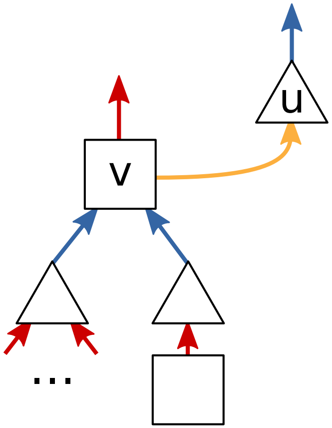
Classification of Profitable Edges

We classify profitable edges by their position in the tree.

Classification of Profitable Edges

We classify profitable edges by their position in the tree.

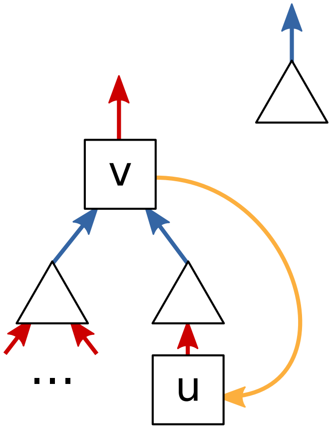
- An edge (v, u) is a cross edge if u is not in the subtree of v.



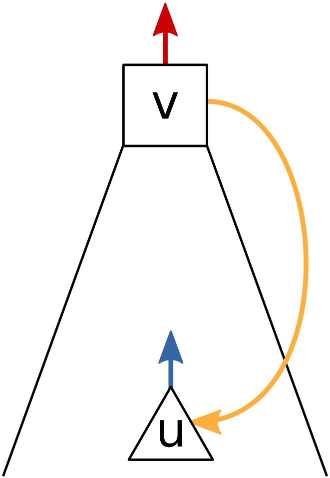
Classification of Profitable Edges

We classify profitable edges by their position in the tree.

- An edge (v, u) is a cross edge if u is not in the subtree of v.
- An edge (v, u) is a back edge if u is in the subtree of v.

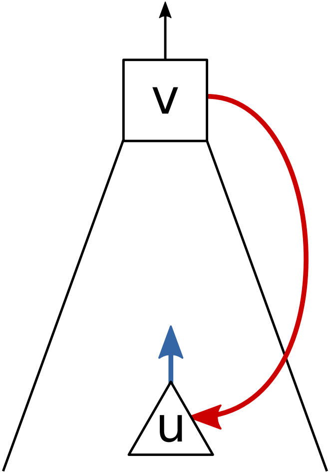


Back Edges



What happens when a profitable back edge is switched?

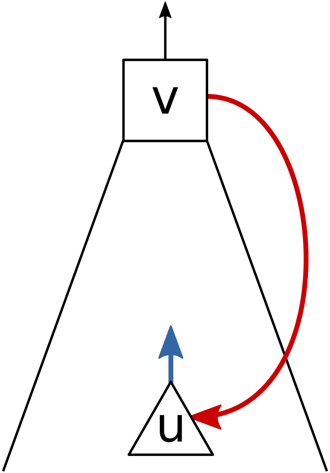
Back Edges



What happens when a profitable back edge is switched?

First Max switches v, disconnecting the subtree of v from the sink.

Back Edges



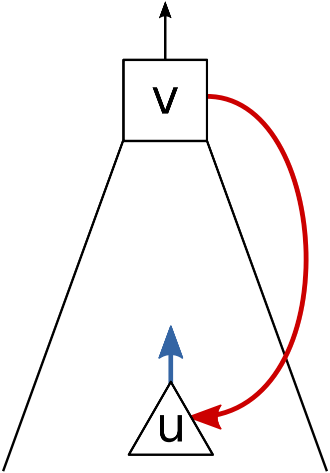
What happens when a profitable back edge is switched?

First Max switches v, disconnecting the subtree of v from the sink.

Then a new best response is computed:

- If the best response does not reconnect a vertex then the valuation must rise to ∞.
- If a vertex is reconnected, then it will have a finite valuation.

Back Edges



What happens when a profitable back edge is switched?

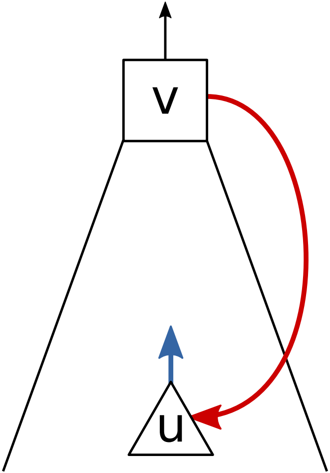
First Max switches v, disconnecting the subtree of v from the sink.

Then a new best response is computed:

- If the best response does not reconnect a vertex then the valuation must rise to ∞.
- If a vertex is reconnected, then it will have a finite valuation.

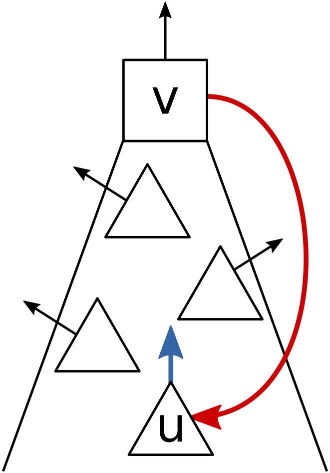
The strategy is winning for the subgame induced by the subtree of v.

Back Edges



How can vertices be reconnected to the sink?

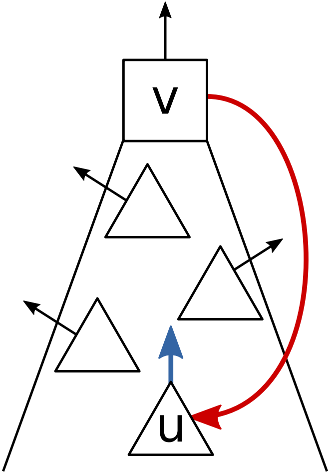
Back Edges



How can vertices be reconnected to the sink?

An escape edge is an edge that Min can use to leave the subtree of v.

Back Edges



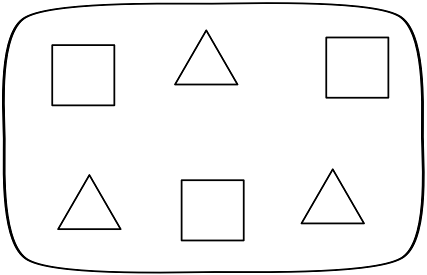
How can vertices be reconnected to the sink?

An escape edge is an edge that Min can use to leave the subtree of v.

To reconnect the subtree with the sink, the best response must use at least one escape edge.

Remembering Back Edges

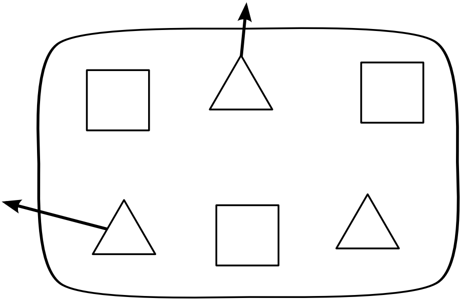
Remembering Back Edges



For each profitable back edge we record:

- the set of vertices in the subtree of v.

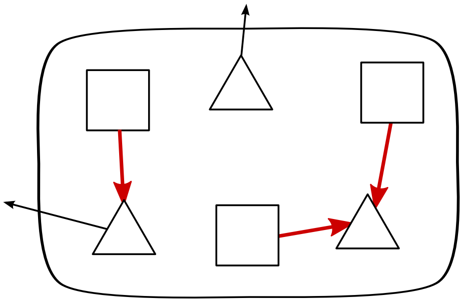
Remembering Back Edges



For each profitable back edge we record:

- the set of vertices in the subtree of v.
- the set of escapes

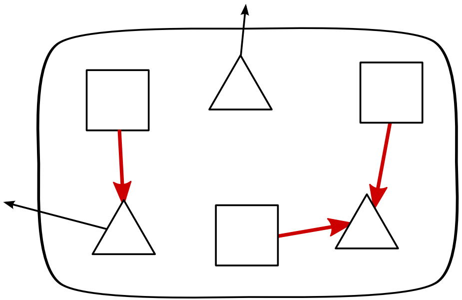
Remembering Back Edges



For each profitable back edge we record:

- the set of vertices in the subtree of v.
- the set of escapes
- the current Max strategy, with v switched to u.

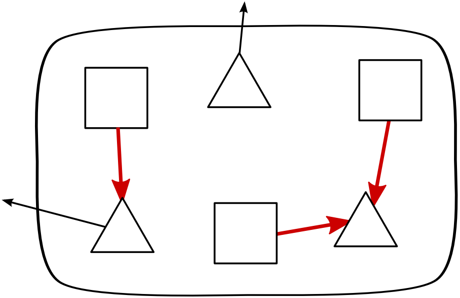
Remembering Back Edges



For each profitable back edge we record:

- the set of vertices in the subtree of v.
- the set of escapes
- the current Max strategy, with v switched to u.
  - This strategy is winning for the subtree of v.

Remembering Back Edges



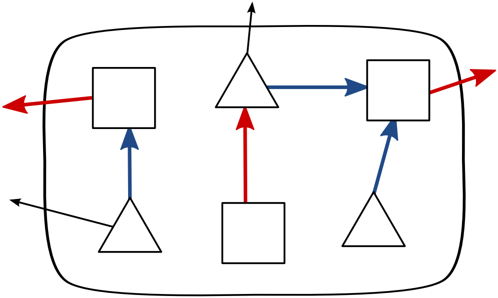
For each profitable back edge we record:

- the set of vertices in the subtree of v.
- the set of escapes
- the current Max strategy, with v switched to u.
  - This strategy is winning for the subtree of v.

We call this a snare.

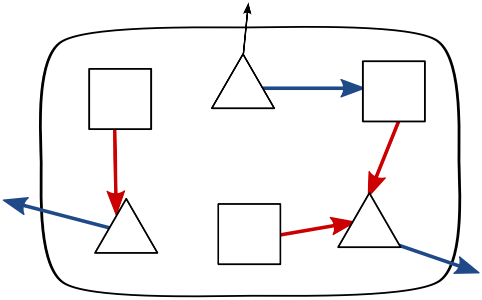
Using Snares

Suppose that we have a strategy σ such that $\mathrm{br}(\sigma)$ does not use an escape from the snare.



Using Snares

Suppose that we have a strategy σ such that $\text{br}(\sigma)$ does not use an escape from the snare.

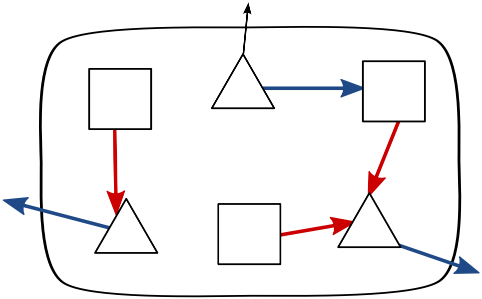


We have a procedure FixSnare which produces a strategy σ' with:

- some escape edge is chosen by $\text{br}(\sigma')$

Using Snares

Suppose that we have a strategy σ such that $\text{br}(\sigma)$ does not use an escape from the snare.

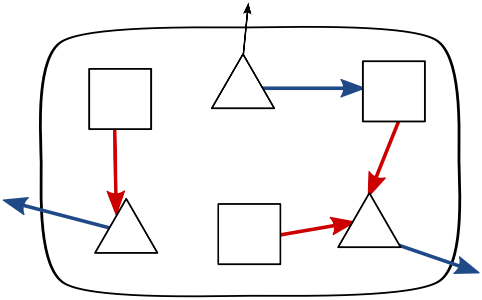


We have a procedure FixSnare which produces a strategy σ' with:

- some escape edge is chosen by $\text{br}(\sigma')$
- σ' is better than σ

Using Snares

Suppose that we have a strategy σ such that br(σ) does not use an escape from the snare.



We have a procedure FixSnare which produces a strategy σ' with:

- some escape edge is chosen by br(σ')
- σ' is better than σ

FixSnare terminates in polynomial time.

How does this help?

What is the advantage of using this procedure?

How does this help?

What is the advantage of using this procedure?

FixSnare can produce a strategy that is better than the strategy obtained by switching an optimal subset of profitable edges!

Non-oblivious Strategy Improvement

while σ has a profitable edge **do**

Compute the best response $\mathsf{br}(\sigma)$.
Compute $\mathsf{Val}^{\sigma,\mathsf{br}(\sigma)}(v)$ for every vertex v.
Compute the set of profitable edges in σ.

$\sigma := \sigma$ with some subset of profitable edges switched.

end while

Non-oblivious Strategy Improvement

while σ has a profitable edge **do**

  $\texttt{Snares} := \texttt{Snares} \cup$ snares for all profitable back edges in σ.

  Compute the best response $\text{br}(\sigma)$.

  Compute $\text{Val}^{\sigma,\text{br}(\sigma)}(v)$ for every vertex v.

  Compute the set of profitable edges in σ.

  either

    $\sigma := \sigma$ with some subset of profitable edges switched.

  or

    $\sigma :=$ FixSnare(σ, S) for some S in $\texttt{Snares}$.

end while

Non-oblivious Strategy Improvement

Friedmann's examples force oblivious switching policies to switch an exponential number of profitable back edges.

Non-oblivious Strategy Improvement

Friedmann's examples force oblivious switching policies to switch an exponential number of profitable back edges.

However, there are only linearly many snares in the example.

Non-oblivious Strategy Improvement

Friedmann's examples force oblivious switching policies to switch an exponential number of profitable back edges.

However, there are only linearly many snares in the example.

Theorem

Non-oblivious strategy improvement terminates in polynomial time on Friedmann's examples.

Conclusions

Non-oblivious strategy improvement is not vulnerable to techniques used to show exponential time behaviour for oblivious strategy improvement.

Conclusions

Non-oblivious strategy improvement is not vulnerable to techniques used to show exponential time behaviour for oblivious strategy improvement.

Future Work

- Prove good complexity bounds for non-oblivious strategy improvement...

Conclusions

Non-oblivious strategy improvement is not vulnerable to techniques used to show exponential time behaviour for oblivious strategy improvement.

Future Work

- Prove good complexity bounds for non-oblivious strategy improvement...
- or find an exponential time example.