An Empirical Analysis of Algorithms for Simple Stochastic Games

Cody Klingler and K. Subramani.

Presented by: Luke Hawranick

November 14, 2024

Lane Department of Computer Science and Electrical Engineering
West Virginia University

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Stochastic Games

Stochastic Games

An omniscient adversary and a surveillance drone move between targets on a graph.

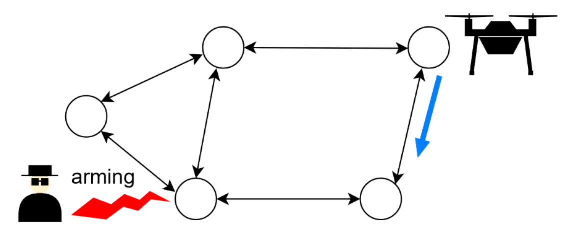


Figure 1: Stackelberg Surveillance

A **stochastic game** is a repeated game with probabilistic transitions played by 2 or more players.

Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V, E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

Rules of an *SSG* - Graph Initialization

A Simple Stochastic Game

Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V, E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

Rules of an ***SSG*** **- Graph Initialization**

- $SINK = \{\text{0-sink}, \text{1-sink}\}$

A Simple Stochastic Game

Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V, E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

Rules of an *SSG* - Graph Initialization

- $SINK = \{\text{0-sink}, \text{1-sink}\}$
- If $v \notin SINK$, $d^+(v) = 2$. Otherwise, $d^+(v) = 0$.

Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V, E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

Rules of an *SSG* - Players

A Simple Stochastic Game

Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V, E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

Rules of an *SSG* - Players

- There are 2 players of the game: MIN and MAX.

A Simple Stochastic Game

Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V, E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

Rules of an *SSG* - Players

- There are 2 players of the game: MIN and MAX.
- At the start of the game, a token is placed on some start vertex. At each step of the game, the token is passed between neighboring vertices.

A Simple Stochastic Game

Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V,E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

Rules of an *SSG* - Players

- There are 2 players of the game: MIN and MAX.
- At the start of the game, a token is placed on some start vertex. At each step of the game, the token is passed between neighboring vertices.
- MIN plays on $v \in V_{MIN}$ and MAX plays on V_{MAX}.

A Simple Stochastic Game

Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V, E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

Rules of an *SSG* **- Players**

- There are 2 players of the game: *MIN* and *MAX*.
- At the start of the game, a token is placed on some start vertex. At each step of the game, the token is passed between neighboring vertices.
- *MIN* plays on $v \in V_{MIN}$ and *MAX* plays on V_{MAX}.
- At $v \in V_{AVE}$, the movement of the token is determined uniformly at random.

A Simple Stochastic Game

Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V, E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

Rules of an SSG **- Players**

- There are 2 players of the game: MIN and MAX.
- At the start of the game, a token is placed on some start vertex. At each step of the game, the token is passed between neighboring vertices.
- MIN plays on $v \in V_{MIN}$ and MAX plays on V_{MAX}.
- At $v \in V_{AVE}$, the movement of the token is determined uniformly at random.
- Players adhere to a strategy that is fixed before the start of the game. A strategy consists of a single edge outgoing from each of a player's vertices.

A Simple Stochastic Game

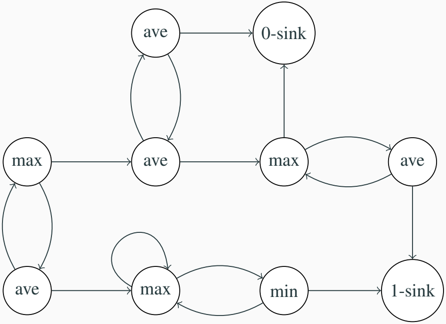
Simple Stochastic Game

A Simple Stochastic Game (SSG), G is a two-player game, defined on a directed multigraph $G(V, E)$. The vertex set V is partitioned into disjoint subsets V_{MAX}, V_{MIN}, V_{AVE}, and $SINK$.

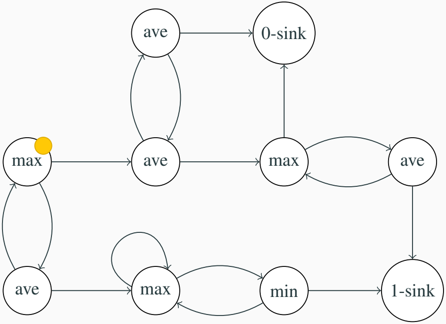
Rules of an *SSG* - Players

- There are 2 players of the game: *MIN* and *MAX*.
- At the start of the game, a token is placed on some start vertex. At each step of the game, the token is passed between neighboring vertices.
- *MIN* plays on $v \in V_{MIN}$ and *MAX* plays on V_{MAX}.
- At $v \in V_{AVE}$, the movement of the token is determined uniformly at random.
- Players adhere to a strategy that is fixed before the start of the game. A strategy consists of a single edge outgoing from each of a player's vertices.
- **The game ends when the token arrives at a *SINK* vertex**. If it arrives at 1-sink, *MAX* wins. Otherwise, *MIN* wins.

Example 1 - A win for *MAX*

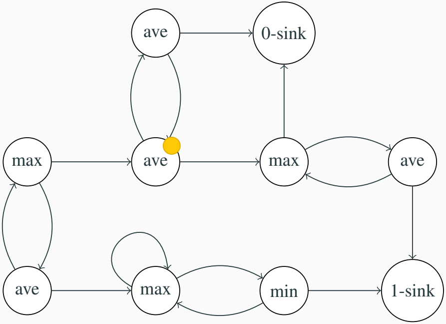


Example 1 - A win for *MAX*



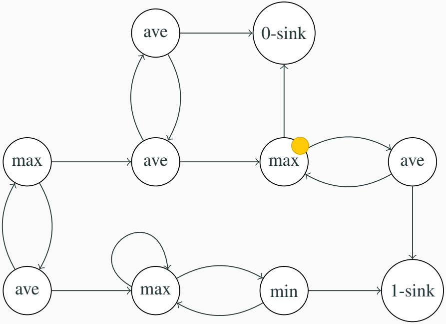
A Simple Stochastic Game

Example 1 - A win for *MAX*



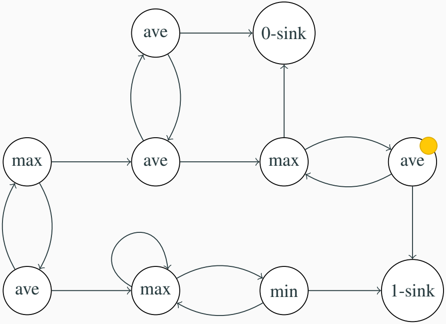
A Simple Stochastic Game

Example 1 - A win for *MAX*



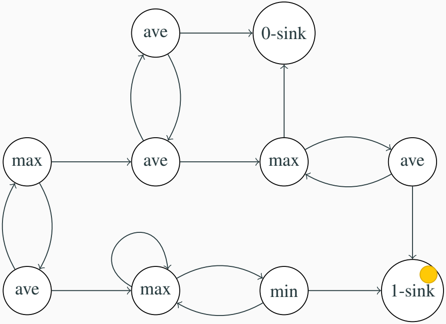
A Simple Stochastic Game

Example 1 - A win for *MAX*



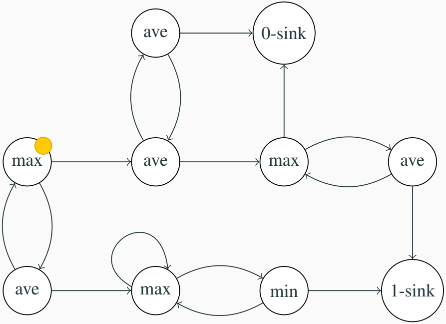
A Simple Stochastic Game

Example 1 - A win for *MAX*



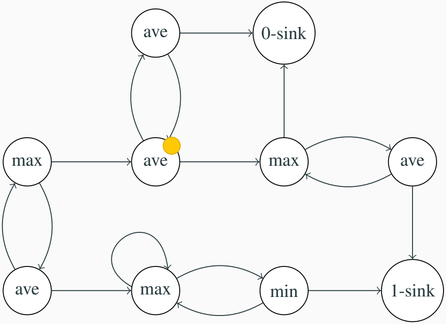
A Simple Stochastic Game

Example 2 - A win for *MIN*

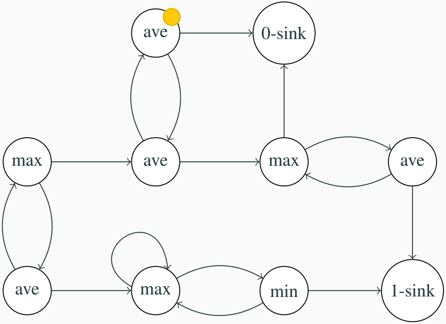


A Simple Stochastic Game

Example 2 - A win for *MIN*

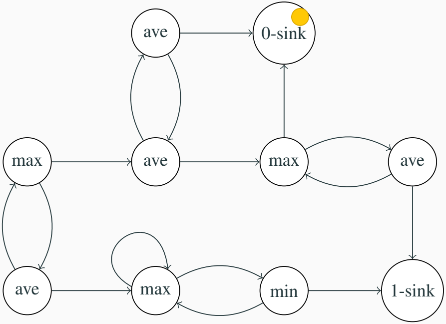


Example 2 - A win for *MIN*



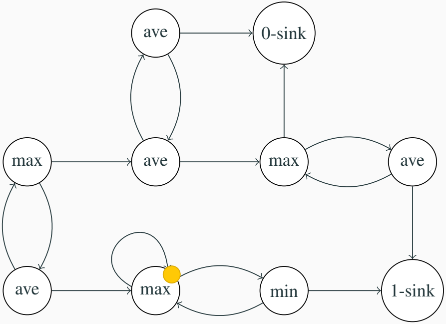
A Simple Stochastic Game

Example 2 - A win for *MIN*



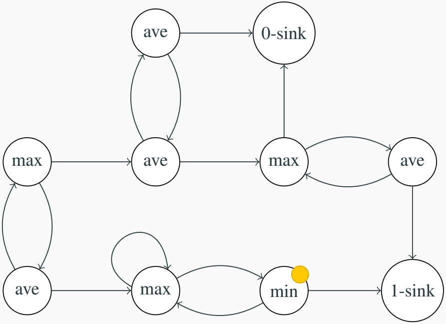
A Simple Stochastic Game

Example 3



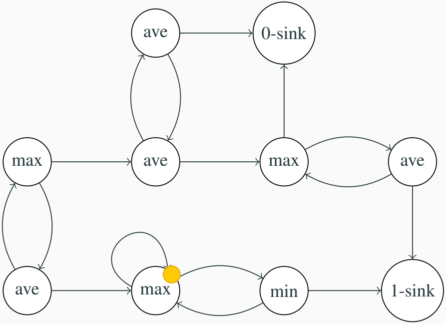
A Simple Stochastic Game

Example 3



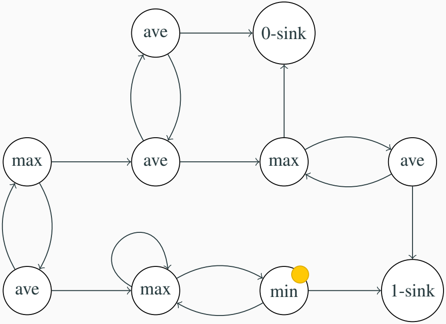
A Simple Stochastic Game

Example 3



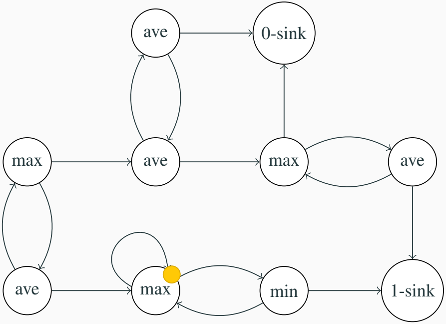
A Simple Stochastic Game

Example 3



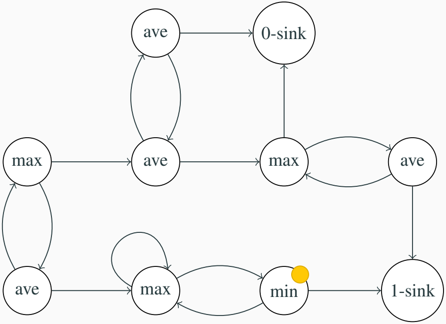
A Simple Stochastic Game

Example 3



A Simple Stochastic Game

Example 3



Complexity and Theoretical Motivations

Probability Decision Problem

Given an SSG with a particular starting vertex, is there a strategy for player *MAX* that guarantees victory with a probability of at least $\frac{1}{2}$ regardless of the strategy of *MIN*?

Complexity and Theoretical Motivations

Probability Decision Problem

Given an SSG with a particular starting vertex, is there a strategy for player *MAX* that guarantees victory with a probability of at least $\frac{1}{2}$ regardless of the strategy of *MIN*?

This problem is in $\mathbf{NP} \cap \mathbf{coNP}$ with no known solution in $\mathbf{P}$.

Complexity and Theoretical Motivations

Probability Decision Problem

Given an SSG with a particular starting vertex, is there a strategy for player *MAX* that guarantees victory with a probability of at least $\frac{1}{2}$ regardless of the strategy of *MIN*?

This problem is in $\mathbf{NP} \cap \mathbf{coNP}$ with no known solution in $\mathbf{P}$.

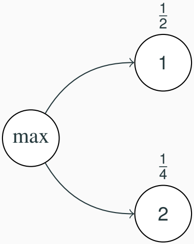
Given the graph G, the most common approach is to compute the optimal strategies for both players.

Strategies and Probability

Propogating Probabilities

V_{MAX}, V_{MIN} **Probabilities**

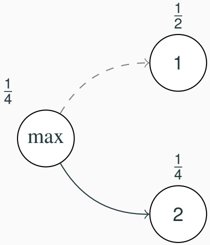
The probability of player *MAX*'s victory for *max* and *min* vertices with one outgoing edge is equal to the probability of victory at the vertex connected by that edge.



Propogating Probabilities

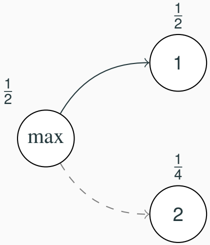
V_{MAX}, V_{MIN} Probabilities

The probability of player *MAX*'s victory for *max* and *min* vertices with one outgoing edge is equal to the probability of victory at the vertex connected by that edge.



Propogating Probabilities

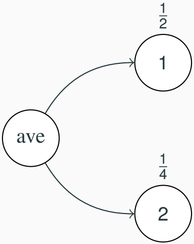
V_{MAX}, V_{MIN} Probabilities
The probability of player *MAX*'s victory for *max* and *min* vertices with one outgoing edge is equal to the probability of victory at the vertex connected by that edge.



Propogating Probabilities

V_{AVE} **Probabilities**

The probability of player *MAX*'s victory for *ave* vertices is equal to the average probability of the vertices connected by the outgoing edges.



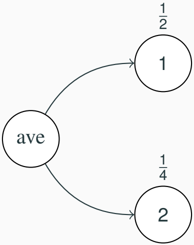
Propogating Probabilities

V_{AVE} **Probabilities**

The probability of player *MAX*'s victory for *ave* vertices is equal to the average probability of the vertices connected by the outgoing edges.

$$v(ave) = \frac{1}{2}v(1) + \frac{1}{2}v(2)$$

$$v(ave) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{4} = \frac{3}{8}$$

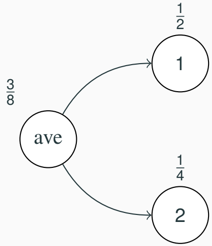


Propogating Probabilities

V_{AVE} **Probabilities**

The probability of player *MAX*'s victory for *ave* vertices is equal to the average probability of the vertices connected by the outgoing edges.

$$v(ave) = \frac{1}{2}v(1) + \frac{1}{2}v(2)$$
$$v(ave) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{4} = \frac{3}{8}$$

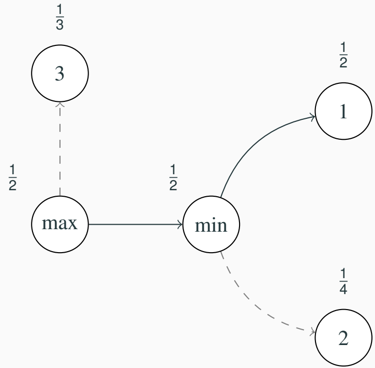


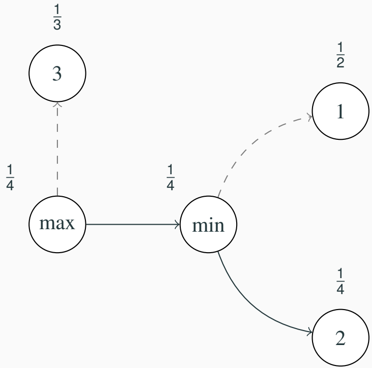
Optimal Strategies

$v \in V_{MAX} \cup V_{MIN}$ is $\tilde{v}$**-switchable** iff $v(i)$ is not locally optimal.

Optimal Strategies

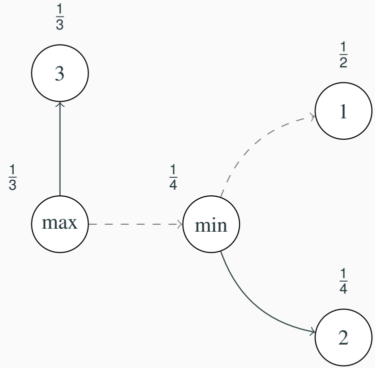
$v \in V_{MAX} \cup V_{MIN}$ is $\tilde{v}$**-switchable** iff $v(i)$ is not locally optimal. If a vertex is not $\tilde{v}$-switchable, then it is $\tilde{v}$*-stable*.


$\frac{1}{3}$
3
$\frac{1}{2}$
1
$\frac{1}{2}$
$\frac{1}{2}$
max
min
$\frac{1}{4}$
2



1/3
3
1/2
1
1/4
1/4
max
min
1/4
2

Now, both the min and max vertices are $\tilde{v}$-stable, meaning that the current strategies are optimal.



Optimal Strategies

A probability vector is considered stable if all player vertices are $\tilde{v}$-stable.

The strategy pair σ, τ is considered *optimal* if it results in the stable vector.

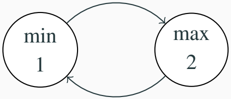
Computing Probabilities

Given σ, τ, the following function computes $v(i)$: the corresponding probability at vertex i, generating a linear system.

$$v_{\sigma,\tau}(i) = \begin{cases} \frac{1}{2}(v_{\sigma,\tau}(j) + v_{\sigma,\tau}(k)) & \text{if } i \in V_{AVE} \text{ for } G_{\sigma,\tau} \text{ with outgoing edges } (i,j), (i,k). \\ v_{\sigma,\tau}(j) & \text{if } i \in V_{MAX} \text{ or } i \in V_{MIN} \text{ for } G_{\sigma,\tau} \text{ with outgoing edge } (i,j). \\ 0 & \text{if } i \text{ is the 0-sink} \\ 1 & \text{if } i \text{ is the 1-sink} \end{cases}$$

Edge Case

Consider the linear system produced by the following vertices: $v(1) = v(2)$

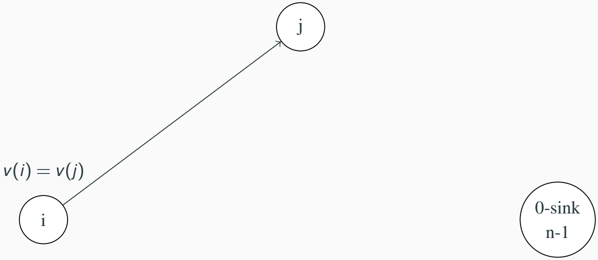


The system is satisfied by any values for the vertices so long as they are the same, but they should have value 0 by the definition of the SSG.

To address this issue, we can reduce an SSG to one where the token always reaches a sink.

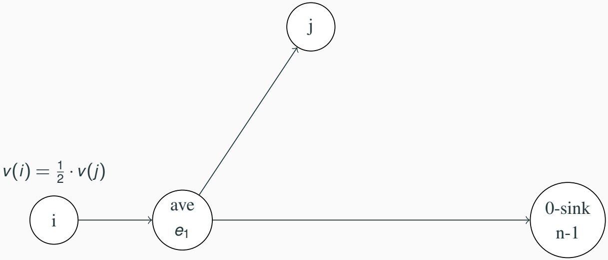
Reduction to a Stopping Game

By replacing all edges between two vertices of $V_{MAX} \cup V_{MIN}$ in an SSG with a chain of *ave* vertices, it may become a stopping game with probabilities arbitrarily close to that of the original game.

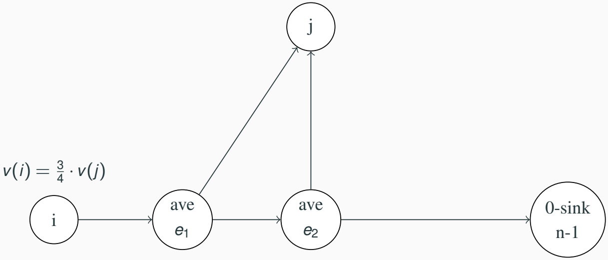


Reduction to a Stopping Game

By replacing all edges between two vertices of $V_{MAX} \cup V_{MIN}$ in an SSG with a chain of *ave* vertices, it may become a stopping game with probabilities arbitrarily close to that of the original game.



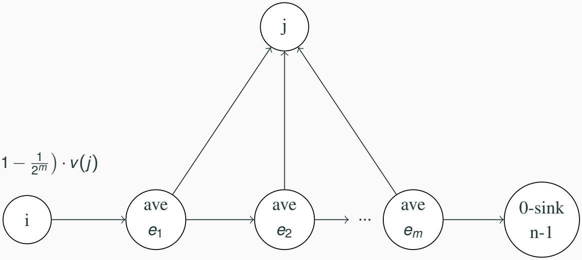
By replacing all edges between two vertices of $V_{MAX} \cup V_{MIN}$ in an SSG with a chain of *ave* vertices, it may become a stopping game with probabilities arbitrarily close to that of the original game.



Reduction to a Stopping Game

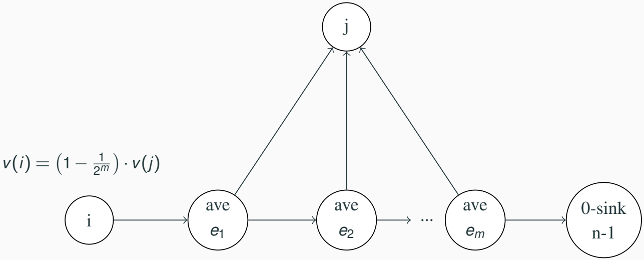
By replacing all edges between two vertices of $V_{MAX} \cup V_{MIN}$ in an SSG with a chain of *ave* vertices, it may become a stopping game with probabilities arbitrarily close to that of the original game.

$v(i) = \left(1 - \frac{1}{2^m}\right) \cdot v(j)$



By replacing all edges between two vertices of $V_{MAX} \cup V_{MIN}$ in an SSG with a chain of *ave* vertices, it may become a stopping game with probabilities arbitrarily close to that of the original game.

Each movement of the token has a probability of at least $\beta = 1/2^m$ of moving to the 0-sink.



Indirectly Reducing to a Stopping Game

Instead of adding numerous ave vertices to introduce the small probability β, the piece-wise function used previously may be modified.

$$v_{\sigma,\tau}(i) = \begin{cases} \frac{1}{2}(v_{\sigma,\tau}(j) + v_{\sigma,\tau}(k)) & \text{if } i \in V_{AVE} \text{ for } G_{\sigma,\tau} \text{ with outgoing edges } (i,j), (i,k). \\ v_{\sigma,\tau}(j) & \text{if } i \in V_{MAX} \text{ or } i \in V_{MIN} \text{ for } G_{\sigma,\tau} \text{ with outgoing edge } (i,j). \\ 0 & \text{if } i \text{ is the 0-sink} \\ 1 & \text{if } i \text{ is the 1-sink} \end{cases}$$

Indirectly Reducing to a Stopping Game

Instead of adding numerous ave vertices to introduce the small probability β, the piece-wise function used previously may be modified.

$$v_{\sigma,\tau}(i) = \begin{cases} \frac{1}{2}((1-\beta)\cdot v_{\sigma,\tau}(j) + (1-\beta)\cdot v_{\sigma,\tau}(k)) & \text{if } i \in V_{AVE} \text{ for } G_{\sigma,\tau} \text{ with outgoing edges } (i,j), (i,k). \\ (1-\beta)\cdot v_{\sigma,\tau}(j) & \text{if } i \in V_{MAX} \text{ or } i \in V_{MIN} \text{ for } G_{\sigma,\tau} \text{ with outgoing edge } (i,j). \\ 0 & \text{if } i \text{ is the 0-sink} \\ 1 & \text{if } i \text{ is the 1-sink} \end{cases}$$

Overview of Algorithms

Finding an Optimal Response

The following two algorithms are used to find stable (optimal) responses, given a fixed strategy for their opponent.

1. Derman's LP
2. Naive Stable Response

Finding a stable vector

The following algorithms will return a stable vector of strategies given a graph G. Three are deterministic:

- Hoffman-Karp
- Non-Convex Quadratic Program (Condon, 1990)
- Converge-From-Below (Condon, 1992)

Two are randomized:

- Tripathi Algorithm (Tripathi, Valkanova, Kumar, 2010)
- Ludwig Algorithm (Ludwig, 1995)

Results

Implementation

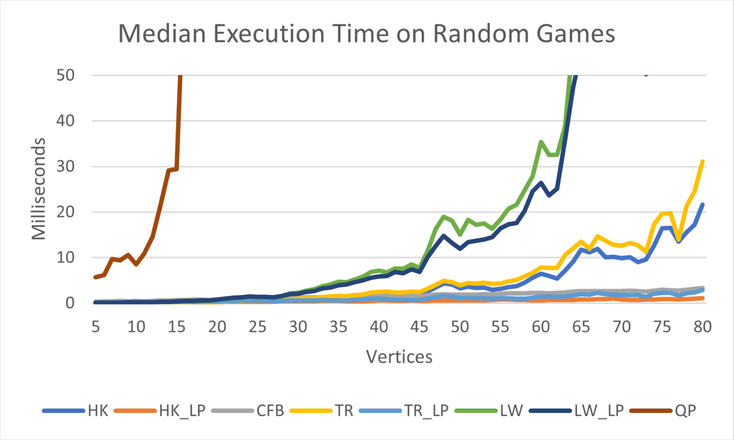
Each of the algorithms discussed were implemented in C++ using the tools Gurobi and Eigen.

Implementation

Each of the algorithms discussed were implemented in C++ using the tools Gurobi and Eigen.

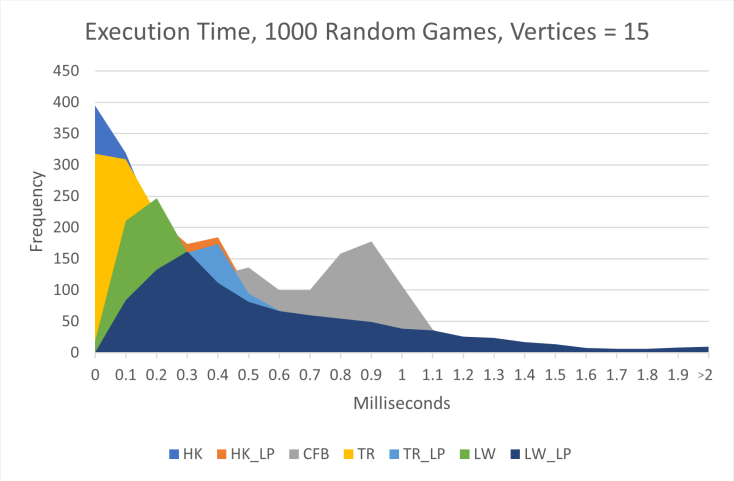
The algorithms that compute the optimal response as an intermediate step were split into two variations, one using Derman's LP and the other using the naive approach.

Empirical Results



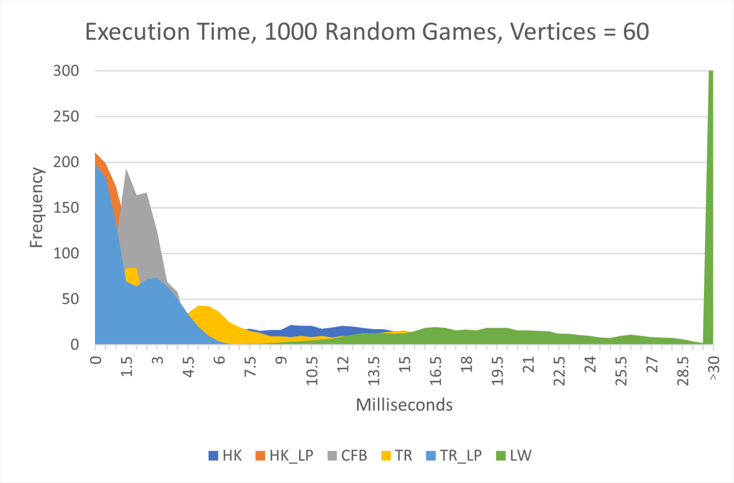
- Quadratic Program is incomparable.
- Converge-From-Below, Tripathi with Derman's LP, and Hoffman-Karp with Dermans' LP are contenders.
- Algorithms with Derman tend to have smaller median execution time compared to those with Naive stable response.

Empirical Results



- On a small instance, all algorithms except QP have comparable runtimes.
- Converge-From-Below has a small spike at a longer execution time.

Empirical Results



- On a larger instance, Ludwig's algorithm becomes incomparable

Searching for Difficult Cases

The Hoffman-Karp algorithm is among the best performing, but its worst case is not known.

Searching for Difficult Cases

The Hoffman-Karp algorithm is among the best performing, but its worst case is not known.

We counted the iterations needed by the Hoffman-Karp algorithm on

- Hundreds of thousands of random games
- Random games without easily solved vertices
- Games found by locally searching the problem space
- Every possible game up 12 vertices

Max-chain Case

The Hoffman-Karp algorithm is among the best performing, but its worst case is not known.

Max-chain Case

The Hoffman-Karp algorithm is among the best performing, but its worst case is not known.

The max-chain only takes $n-1$ iterations of the Hoffman-Karp algorithm to solve.

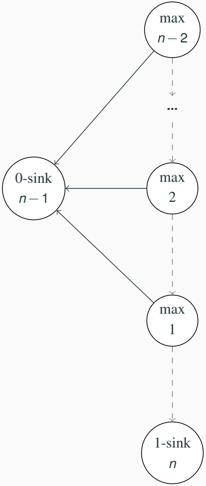


Figure 2: Initialization of max-chain

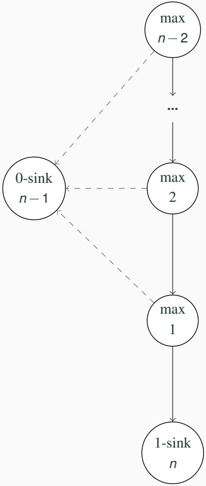
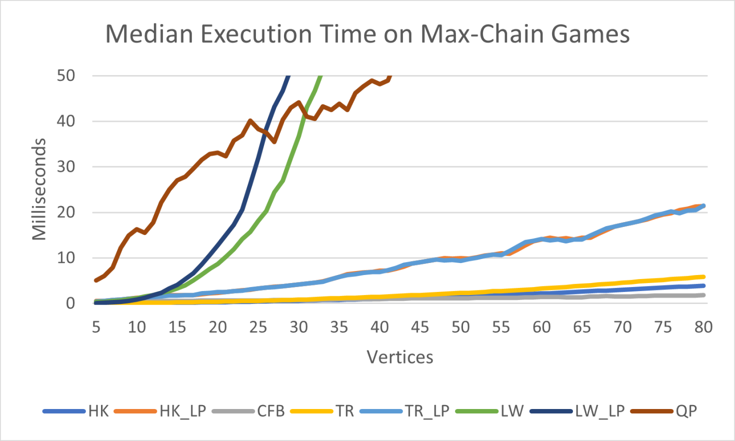


Figure 3: Solved max-chain

Max-chain Case



- On this instance, Ludwig's algorithm performs very badly. QP overcomes it.
- Hoffman-Karp with naive stable response, Tripathi's algorithm, and Converge-From-Below champion this instance.

Modifying the max-chain to include a chain of min vertices yields the following game:

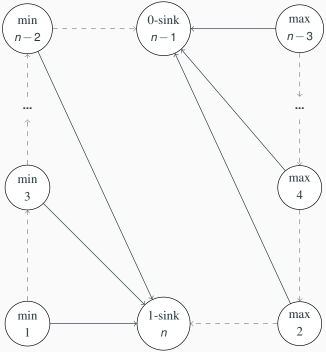


Figure 4: Double-Chain initialization

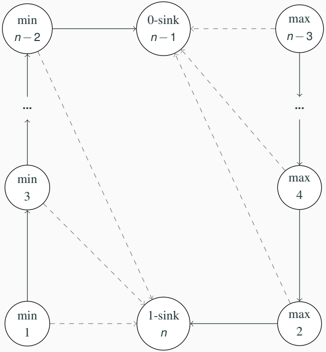
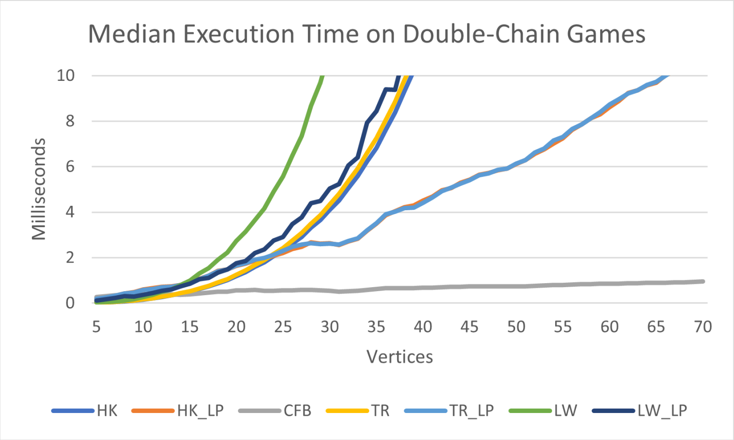


Figure 5: Solved Double-Chain

Double-chain Case



- Converge-From-Below champions this instance.

Conclusion

- The Converge-From-Below algorithm most consistently performs well for the game sizes considered despite being thought to be exponential in the worst case.

Conclusion

- The Converge-From-Below algorithm most consistently performs well for the game sizes considered despite being thought to be exponential in the worst case.
- Many other exponential algorithms, including those by Hoffman-Karp and Tripathi, also perform well in these cases.

Conclusion

- The Converge-From-Below algorithm most consistently performs well for the game sizes considered despite being thought to be exponential in the worst case.
- Many other exponential algorithms, including those by Hoffman-Karp and Tripathi, also perform well in these cases.
- This paper hints that the current complexity results for these algorithms warrant further research.