Solving Simple Stochastic Games

Krishnendu Chatterjee

Tobias Meggendorfer

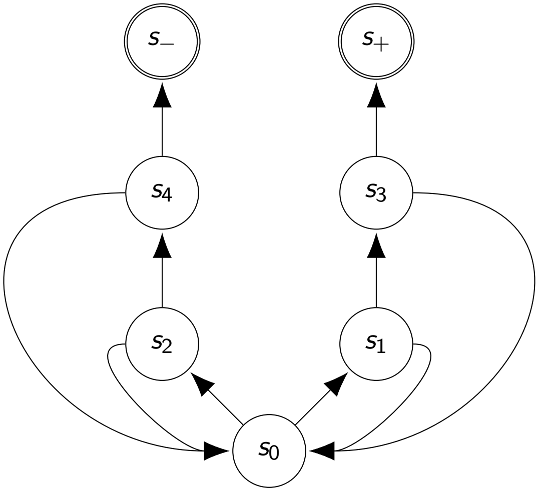
Raimundo Saona

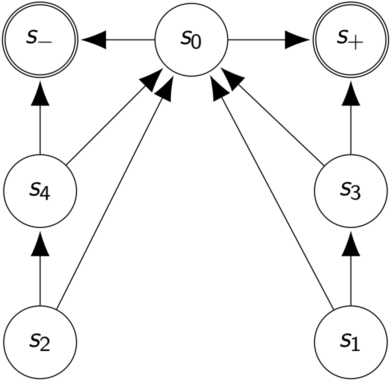
Jakub Svoboda

Based on [CMSS23]

Institute of Science and Technology Austria

Can we **check**
that we do not **lose**
by making **simplifications**?


s_-
s_+
s_4
s_3
s_2
s_1
s_0


s_-
s_0
s_+
s_4
s_3
s_2
s_1

Key concepts

- Simple Stochastic Games
- Parametrized Complexity of Graphs
- Guessing a Value and Verifying the guess

Simple Stochastic Games

Simple Stochastic Games (SSG)

An SSG is a tuple $\Gamma = (S, E, \delta, (S_{\max}, S_{\min}, S_p, \{s_+, s_-\}))$ with the following components.

- The state space $S = S_{\max} \cup S_{\min} \cup S_p \cup \{s_+, s_-\}$, partitioned into player-max's states, player-min states, probabilistic states, and two special absorbing (or sink) states.
- The probabilistic transition function $\delta \colon S_p \to \Delta(S)$.
- The edge set $E \subseteq S \times S$ such that
  - $E(s) := \{s' \mid (s, s') \in E\}$
  - $E(s)$ is non-empty for all states;
  - for all $s \in S_p$ we have $E(s) = \{s' \mid \delta(s, s') > 0\}$;
  - $E(s_+) = \{s_+\}$ and $E(s_-) = \{s_-\}$.

We associate the graph $G = (S, E)$.

Value

- Strategies.

$$\sigma \colon S_{\max} \to S, \quad \sigma(s) \in E(s)$$
$$\tau \colon S_{\min} \to S, \quad \tau(s) \in E(s)$$

- Reachability.

$$\mathbb{P}_s^{\pi=(\sigma,\tau)}(\exists t \geq 0 : S_t = s_+)$$

- Value.

$$\mathrm{val}(s) \coloneqq \max_{\sigma} \min_{\tau} \mathbb{P}_s^{\pi}(\mathrm{Reach}(s_+))$$

Complexity

- Computing the value is in NP and coNP.
- All problems in NP inter coNP have eventually fallen into P.
- All known algorithms run in exponential time worst case.
- There is a randomized sub-exponential algorithm, proposed in 1995.

Question

Can we get a sub-exponential time deterministic algorithm for a large class of games?

Hard instances

What makes a Simple Stochastic Game hard?

Parametrized Complexity

Parametrized Complexity

Some notions for the complexity of graphs are the following.

Definition (Tree-depth)

Shortest tree where all edges are between nodes in an ancestor-descendent relationship.

Definition (Cycle rank)

Minimum number of vertices that one must remove to make the graph acyclic.

Definition (Tree-width)

Minimum size of bags, where every edge is between nodes of the same bag.

Complexity considerations

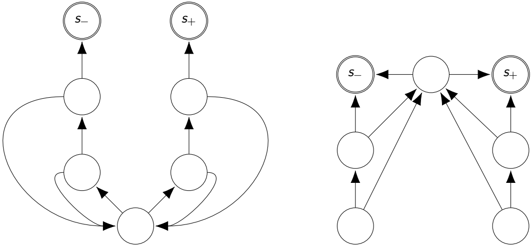
Consider the tree-width.

- It is NP-complete to compute the tree-width of any graph.
- For a fixed tree-width t, it is linear time to recognize a graph with tree-width t and compute its tree decomposition.

This is the limit of our fixed-parameter complexity.

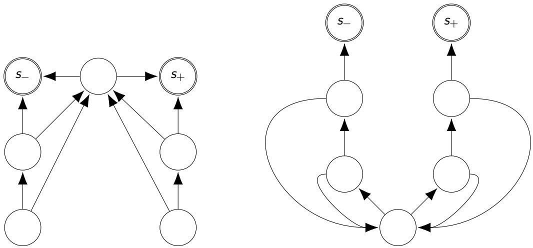
Simplification: Guessing

Guessing



- Pick a state and a guess of its value.
- Replace all outgoing edges by just two, going to s_- and s_+ with corresponding probabilities.

Verification



- After a guess is made, take back the graph transformation.
- Apply a local update in the state and compare it with the guess.

Verification: Formal statement

Lemma

Consider a game G, a state $s \in S$ and a guess $\gamma \in [0, 1]$.

$$\mathrm{Update}(s, (\gamma, \mathsf{val}_{G[s=\gamma]})) > \gamma \quad \Leftrightarrow \quad \mathsf{val}_G(s) > \gamma \, .$$

Remark

Verification does not only give you true or false but allows binary search the value.

Approximating by guessing

- Binary search the value of a state, starting with bounds $[0, 1]$ and solving **exactly** the guessed game $G[s = \gamma]$.
- After $\log(1/\varepsilon)$ iterations, you obtained an ε-approximation of the value.

Question

How to transform an approximate solution into an exact solution?

From Approximation to Exact

Definition (Value separation)

Given a game G, the value separation is

$$B := \min\{|\mathrm{val}(s) - \mathrm{val}(s')| : \mathrm{val}(s) \neq \mathrm{val}(s')\} .$$

To retrieve an exact solution from an approximate solution,

- Compute $(B/2)$-approximations in every state.
- Construct a strategy for each player based on the current ranking of the values.
- Since this strategy is optimal, compute the values.

Value Separation for SSGs

Definition (Transition complexity)

Given a game G, the transition complexity for a probabilitistic state s is

$$D(s) := \min\{M : \forall s' \in E(s) : M\delta(s, s') \in \mathbb{N}\} .$$

Then, the overall transition complexity is

$$D := \max\{D(s) : s \in S\} .$$

Lemma

Consider a game G with state space S and rational transition probabilities with transitions complexity D. Then, the value separation is at least $1/(2D)^{|S|-1}$.

Summary for guessing a state

- Pick a state s to apply guessing.
- Guess it and solve the resulting game $G[s = \gamma]$.
- Repeat until the approximation of the $\mathrm{val}(s)$ is sufficiently good.
- Compute $\mathrm{val}(s)$ exactly from the approximation and the solution of the value of $G[s = \gamma]$.

This gives an algorithm that, given an Oracle to solve a game, solves the game for a larger game.

Final algorithm

Pseudo-algorithm

The main idea is as follows.

- Small tree-width allows you to obtain smaller independent instances of Simple Stochastic Games after guessing a few states.
- By solving these smaller SSGs recursively, one obtains an improved complexity, parametrized by the tree-width.

Result

Theorem

Given a game G with state space S and transitions complexity D whose game graph has tree-width t, there is an algorithm that computes the value vector val_G in time

$$\mathcal{O}\left(\left(t|S|^2 \log D\right)^{t \log |S|}\right).$$

What did we need to make this work?

In general, these are the tools you need.

- Guessing simplifies the problem.
- Guessing can be verified, and the verification is informative.
- With enough information I can give an exact solution.

In our case, this was possible in SSGs because

- A parametrized complexity had not had an efficient solution.
- For a fixed parameter, the guarantees can be computed efficiently.

References I

Krishnendu Chatterjee, Tobias Meggendorfer, Raimundo Saona, and Jakub Svoboda.
Faster Algorithm for Turn-based Stochastic Games with Bounded Treewidth.
In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4590–4605, January 2023.