Solving Simple Stochastic Games

Hugo Gimbert, Florian Horn

Solving Simple Stochastic Games*

Hugo Gimbert[1] and Florian Horn[2]

[1] LaBRI, Université Bordeaux 1, France
hugo.gimbert@labri.fr
[2] LIAFA, Université Paris 7, France
florian.horn@liafa.jussieu.fr

Abstract. We present a new algorithm for solving Simple Stochastic Games (SSGs), which is fixed parameter tractable when parametrized with the number of random vertices. This algorithm is based on an exhaustive search of a special kind of positional optimal strategies, the *f-strategies*. The running time is $\mathcal{O}(\ |V_{\mathrm{R}}|! \cdot (\log(|V|)|E| + |p|)\ )$, where $|V|, |V_{\mathrm{R}}|, |E|$ and $|p|$ are respectively the number of vertices, random vertices and edges, and the maximum bit-length of a transition probability. Our algorithm improves existing algorithms for solving SSGs in three aspects. First, our algorithm performs well on SSGs with few random vertices, second it does not rely on linear or quadratic programming, third it applies to all SSGs, not only stopping SSGs.

Introduction

Simple Stochastic Games (SSGs for short) are played by two players Max and Min in a sequence of steps. Players move a pebble along edges of a directed graph (V, E). There are three type of vertices: V_{Max} is the set of vertices of player Max, V_{Min} the set of vertices of player Min and V_{R} the set of random vertices. When the pebble is on a vertex of V_{Max} or V_{Min}, the corresponding player chooses an outgoing edge and moves the pebble along it. When the pebble is on a random vertex, the successor is chosen randomly according to some fixed probability distribution: from vertex $v \in V_{\mathrm{R}}$ the pebble moves towards vertex $w \in V$ with some probability $p(w|v)$ and the probability that the game stops is 0, *i.e.* $\sum_{w \in V} p(w|v) = 1$. An SSG is depicted on Figure 1, with vertices of V_{Max} represented as $\bigcirc$, vertices of V_{Min} represented as $\square$, and vertices of V_{R} represented as $\Diamond$.

Player Max and Min have opposite goals, indeed player Max wants the pebble to reach a special vertex $t \in V$ called the *target vertex*, if this happens *the play is won by player* Max. In the opposite case, the play proceeds forever without reaching t and is *won by player* Min. For technical reasons, we assume that t is a vertex of player Max and is absorbing. *Strategies* are used by players to choose their moves, a strategy tells where to move the pebble depending on the sequence of previous vertices, *i.e.* the finite path followed by the pebble from

* This research was partially supported by french project ANR "DOTS".

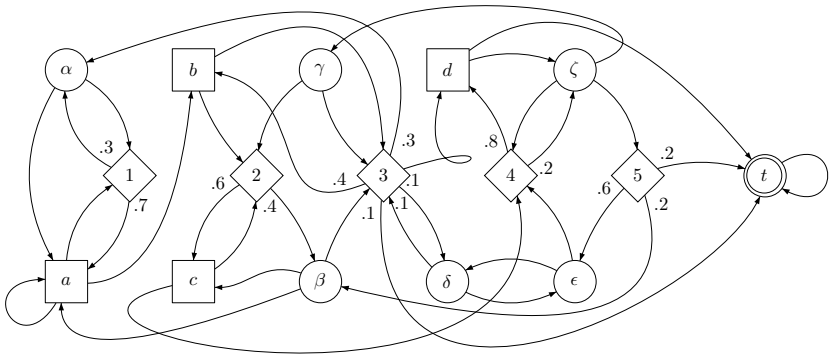

Fig. 1. A Simple Stochastic Game.

the beginning of the play. The *value* of a vertex v is the maximal probability with which player Max can enforce the play to reach the target vertex. When player Max, respectively player Min, uses an *optimal strategy* he ensures reaching the target with a probability greater, respectively smaller, than the value of the initial vertex.

We are interesting in *solving* SSGs, that is computing values and optimal strategies.

Existing algorithms for solving SSGs. The complexity of solving SSGs was first considered by Condon [Con92], who proved that deciding whether the value of an SSG is greater than $\frac{1}{2}$ is in NP $\cap$ co-NP. The algorithm provided in [Con92] consists in first transforming the input SSG in a *stopping SSG* where the probability to reach a sink vertex is 1. The transformation keeps unchanged the fact that the initial vertex has value strictly greater than $\frac{1}{2}$ but induces a quadratic blowup of the size of the SSG. The algorithm then non-deterministically guesses the values of vertices, which are rational numbers of linear size, and checks that these values are the unique solutions of some *local optimality equations*.

Three other kinds of algorithms for solving SSGs are presented in [Con93]. These algorithms require transformation of the initial SSG into an equivalent *stopping* SSG and are based on local optimality equations. First algorithm computes values of vertices using a quadratic program with linear constraints. Second algorithm computes iteratively from below the values of the SSGs, and the third is a strategy improvement algorithm *à la* Hoffman-Karp. These two last algorithms require solving an exponential number of linear programs, as it is the case for the algorithm recently proposed in [Som05].

Finally, these four algorithms suffer three main drawbacks.

First, these algorithms rely on solving either an exponential number of linear programs or a quadratic program, which may have prohibitive algorithmic cost and makes the implementation tedious.

Second, these algorithms only apply to the special class of *stopping* SSGs. Although it is possible to transform any SSG into a stopping SSG with arbitrarily small change of values, computing exact values this way requires to modify drastically the original SSG, introducing either $|V|^2$ new random vertices or new transition probabilities of bit-length quadratic in the original bit-length. This also makes the implementation tedious.

Third, the running time of these algorithms may *a priori* be exponential whatever be the number of random vertices of the input SSG, including the case of SSGs with no random vertices at all, also known as reachability games on graphs. However it is well-known that reachability games on graphs are solvable in linear time.

Notice that randomized algorithms do not perform much better since the best randomized algorithms [Lud95,Hal07] known so far run in sub-exponential expected time $e^{O(\sqrt{n})}$.

Our results. In this paper we present an algorithm that computes values and optimal strategies of an SSG in time $\mathcal{O}(\ |V_{\mathrm{R}}|! \cdot (\log(|V|)|E| + |p|)\ )$, where $|V_{\mathrm{R}}|$ is the number of random vertices, $|V|$ is the number of vertices and $|p|$ is the maximal bit-length of transition probabilities.

The key point of our algorithm is the fact that optimal strategies may be looked for in a strict subset of positional strategies, called the class of $\mathbf{f}$-strategies. The $\mathbf{f}$-strategies are in correspondence with permutations of random vertices. Our algorithm does an exhaustive search of optimal $\mathbf{f}$-strategies among the $|V_{\mathrm{R}}|!$ available ones and check their optimality. Optimality is easy to check, it consists in computing a reachability probability in a Markov Chain with V_{R} states, which amounts to solving a linear system with at most $|V_{\mathrm{R}}|$ equations.

Comparison with existing work. We improve existing results by three aspects: our algorithm performs better on SSGs with few random vertices, it is arguably much simpler, and we provide new insight about the structure of optimal strategies.

Our algorithm performs much better on SSGs with few random vertices than previously known algorithms. Indeed, its complexity is $\mathcal{O}(\ |V_{\mathrm{R}}|! \cdot (\log(|V|)|E| + |p|)\ )$, hence when there are no random vertices at all, our algorithm matches the usual quadratic complexity for solving reachability games on graphs. When the number of random vertices is fixed, our algorithm performs in polynomial time, and on the class of SSGs such that $|V_{\mathrm{R}}| \leq \sqrt{|V_{\mathrm{Max}}| + |V_{\mathrm{Min}}|}$ our algorithm is sub-exponential.

Whereas the complexity is optimal when there are no random vertices, this is no more the case when there are no vertices for player Max or Min. In that case, there exists polynomial time algorithm, whereas the complexity of our algorithm remains exponential in the number of random vertices.

Our algorithm is arguably simpler than previously known algorithms. Indeed, it does not require use of linear or quadratic programming. Although linear programs can be solved in polynomial time [Kac79,Ren88], this requires high-precision arithmetic. By contrast, our algorithm is very elementary: it enu-

merates permutations of the random vertices and for each permutation, it solves a linear system of equations.

Our algorithm is also simpler because it applies directly to any kind of SSGs, whereas previously known algorithms require the transformation of the input SSG into a stopping SSG of quadratic size. As a consequence, we can use the same algorithm for solving other types of infinite duration game; this is ongoing work.

The full paper is available online [GH07].

Acknowledgments We thank an anonymous referee for his very insightful and helpful comments.

References

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[Con93] A. Condon. On algorithms for simple stochastic games. In *Advances in computational complexity theory*, volume 13 of *DIMACS series in discrete mathematics and theoretical computer science*, pages 51–73, 1993.

[Der72] C. Derman. *Finite State Markov Decision Processes*. Academic Press, 1972.

[Dix82] J. D. Dixon. Exact solution of linear equations using p-adic expansions. *Numerische Mathematik*, 40:137–141, 1982.

[GH07] H. Gimbert and F. Horn. Solving simple stochastic games with few random vertices. http://hal.archives-ouvertes.fr/hal-00195914/fr/.

[Hal07] N. Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. *Algorithmica*, 49:37–50, 2007.

[Kac79] L. G. Kachiyan. A polynomial time algorithm for linear programming. *Soviet Math Dokl.*, 20:191–194, 1979.

[Lud95] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.

[Ren88] J. Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40:59–93, 1988.

[Sha53] L. S. Shapley. Stochastic games. In *Proceedings of the National Academy of Science USA*, volume 39, pages 1095–1100, 1953.

[Som05] Rafal Somla. New algorithms for solving simple stochastic games. *Electr. Notes Theor. Comput. Sci.*, 119(1):51–65, 2005.