

# A Subexponential Randomized Algorithm for the Simple Stochastic Game Problem\*

WALTER LUDWIG

Department of Computer Science, University of Wisconsin-Madison, Madison, Wisconsin 53706

---

We describe a randomized algorithm for the simple stochastic game problem that requires  $2^{O(\sqrt{n})}$  expected operations for games with  $n$  vertices. This is the first subexponential time algorithm for this problem.

© 1995 Academic Press, Inc.

---

## 1. INTRODUCTION

A *simple stochastic game* is a directed graph with three types of vertices, *min*, *max*, and *average*, along with two sink vertices, the 0-sink and the 1-sink. Each vertex has two outgoing edges, except for the sink vertices, which have no outgoing edges. One of the vertices is a *start* vertex.

The game is a contest between two players, 0 and 1. It is played in the following way. Begin by placing a token on the start vertex. When the token is on a max vertex  $i$ , player 1 moves it along one of the outgoing edges of  $i$ . When the token is on a min vertex  $j$ , player 0 moves it along one of the outgoing edges of  $j$ . When the token is on an average vertex, the edge along which the token is moved is determined by the toss of a fair coin. The game ends when one of the sink vertices is reached. The goal of player 1 is to reach the 1-sink. The goal of player 0 is to avoid reaching the 1-sink.

Before the game begins, player 1 will choose an outgoing edge from each max vertex. These edges will define a *strategy* for player 1. During the game, whenever the token is on a max vertex, player 1 will move the token along the edge that is included in her strategy. (Defining strategies deterministically in this way does not result in any loss of generality (Condon, 1992a)). Similarly, a choice of an outgoing edge for each min vertex is a strategy for player 0.

Given a simple stochastic game, we would like to find the best strategies for both players. The corresponding decision problem is to determine whether player 1 wins with probability greater than  $\frac{1}{2}$  when both players use their best strategies.

Simple stochastic games are a restriction of the stochastic games introduced by Shapley (1953). In these games, *both* players choose actions independently at each step from finite sets of possible actions available in the current state

(i.e., vertex). Player 1 then receives a (possibly negative) reward depending on the actions of the two players. The players' actions also determine for each state a probability that the game will move to that state in the next step.

Many variations of the general stochastic game problem have been studied since it was first introduced. (See (Peters and Vrieze, 1987) for a survey.) Among these are the switching control games of Filar (1981) and the additive reward and additive transitions games of Raghavan *et al.* (1985), both of which can be viewed as generalizations of the simple stochastic game problem.

The simple stochastic game problem was introduced by Condon (1992a) to model the computation of space-bounded alternating Turing machines that have random as well as universal and existential states. (See (Chandra *et al.*, 1981) for a description of alternating Turing machines.) A deterministic polynomial time algorithm for the simple stochastic game problem would imply that adding randomness to space-bounded alternating Turing machines does not alter the class of languages they accept. But while the simple stochastic game problem is known to belong to the class  $NP \cap co-NP$  (Condon, 1992a), no polynomial time algorithm for it is known. In fact, although many algorithms for this problem have been considered, several of them have been shown to be exponential, and none of them are known to be subexponential. See (Condon, 1992b) for a survey of algorithms for this problem.

In this paper, we describe a randomized algorithm which, for any simple stochastic game with  $n$  vertices, requires an expected number of operations that is subexponential in  $n$ .

The algorithm works in the following way. Start at some strategy  $\sigma$  for player 1. Choose at random an edge in this strategy, and then apply the algorithm recursively to find the best strategy  $\sigma'$  that includes this edge. If  $\sigma'$  is not optimal, then the edge that was chosen is not included in an optimal strategy for player 1, so remove it from  $\sigma'$  and replace it with the other edge that originates at the same vertex. Then set  $\sigma := \sigma'$  and repeat.

Note that if this were done deterministically, it would be exhaustive search with backtracking.

Algorithms for the simple stochastic game problem that involve starting at some strategy for player 1 and improving

\* Supported in part by NSF grant CCR-9024516.

it by changing it at one vertex at a time, such as the one presented here, are closely related to simplex algorithms for linear programming. In fact, if the game consists only of max and average vertices (or only min and average vertices), then a linear program can be constructed such that there is a one-to-one correspondence between basic feasible solutions and strategies (Derman, 1972). However, no such construction is known when the game consists of all three types of vertices. But consider a graph in which each vertex represents a strategy for player 1, and in which a pair of vertices are joined by an edge if the corresponding strategies differ at only one vertex in the game, i.e., a  $d$ -dimensional cube representing all possible strategies for player 1, where  $d$  is the number of max vertices in the game. Associate with each player 1 strategy  $\sigma$  a strategy for player 0 that is the best possible strategy against  $\sigma$ . Then our algorithm moves from one vertex of the cube of strategies to another, improving the player 1 strategy until an optimal strategy is found. This is similar to the way that a simplex algorithm operates on the graph corresponding to the feasible polyhedron of a linear program. In this framework, our algorithm corresponds to a randomized version of a simplex pivot rule proposed by Bland (1977). Recently, Kalai (1992) showed that under favorable conditions, this rule yields a subexponential simplex algorithm for linear programming. Adapting methods used by Kalai, we show that this rule also yields a subexponential algorithm for the simple stochastic game problem.

2. DEFINITIONS AND PRELIMINARY RESULTS

DEFINITION 1. A simple stochastic game is a directed graph  $G = (V, E)$  with the following properties.  $V$  is the disjoint union of  $X, N, A$ , and two special vertices, the 0-sink and the 1-sink. The sets  $X, N$ , and  $A$  are the sets of max, min, and average vertices, respectively. There is one vertex in  $V$  that is called the start vertex. Every vertex in  $V$  has two outgoing edges, except for the sink vertices, which have no outgoing edges.

We will let  $d$  denote the number of max vertices in a game, and we will suppose that the vertices are numbered such that  $X = \{1, \dots, d\}$ . We will further suppose that for each non-sink vertex  $i \in V$ , one of the outgoing edges of  $i$  is labeled 0 and the other is labeled 1. The label of the edge  $(i, j) \in E$  will be denoted by  $l(i, j)$ .

DEFINITION 2. A strategy  $\sigma = (\sigma_1, \dots, \sigma_d)$  for player 1 is a bit vector where the component  $\sigma_i$  indicates the label of an edge originating at vertex  $i$ .

A strategy for player 0 is defined similarly.

Let  $\sigma, \tau$  be a pair of strategies for players 1 and 0. Construct the graph  $G_{\sigma, \tau}$  in the following way. Remove from  $G$  each edge  $(i, j)$  where  $i \in X$  and  $\sigma_i \neq l(i, j)$ . Also remove each edge  $(s, t)$  where  $s \in N$  and  $\tau_s \neq l(s, t)$ . A simple

stochastic game halts with probability 1 if and only if for all pairs of strategies  $\sigma, \tau$ , every vertex in  $G_{\sigma, \tau}$  has a path to a sink vertex. In the results that follow, we restrict our attention to games that halt with probability 1. Lemma 3 below shows that we can do this without loss of generality.

DEFINITION 3. The value of a vertex  $i$  with respect to a pair of strategies  $\sigma, \tau$ , denoted by  $v_{\sigma, \tau}(i)$ , is the probability that player 1 will win the game if the start vertex is  $i$  and the players use strategies  $\sigma$  and  $\tau$ .

DEFINITION 4. We say that a vertex  $i \in X$  with outgoing edges  $(i, j)$  and  $(i, k)$  is stable with respect to a pair of strategies  $\sigma, \tau$  if  $v_{\sigma, \tau}(i) = \max\{v_{\sigma, \tau}(j), v_{\sigma, \tau}(k)\}$ . Similarly, we say that a vertex  $i \in N$  with outgoing edges  $(i, j)$  and  $(i, k)$  is stable with respect to a pair of strategies  $\sigma, \tau$  if  $v_{\sigma, \tau}(i) = \min\{v_{\sigma, \tau}(j), v_{\sigma, \tau}(k)\}$ . We say that a vertex is unstable if it is not stable.

DEFINITION 5. Let  $\sigma, \tau$  be a pair of strategies for players 1 and 0. The strategy  $\tau$  is said to be optimal with respect to  $\sigma$  if every min vertex is stable with respect to  $\sigma, \tau$ .

We will let  $\tau(\sigma)$  denote an optimal strategy for player 0 with respect to the player 1 strategy  $\sigma$ . Optimal strategies for player 1 are defined similarly.

LEMMA 1 (Derman, 1972). Let  $H$  be a simple stochastic game with no max vertices that halts with probability 1. Let  $n$  denote the number of vertices in  $H$ , let the vertices of  $H$  be labeled such that  $V = \{1, \dots, n\}$ , and let the 0-sink and the 1-sink be labeled  $n - 1$  and  $n$ , respectively. Then an optimal strategy for player 0 (with respect to the trivial player 1 strategy) can be found by solving the following linear program:

$$\begin{aligned}
 &\text{maximize } \sum_{i=1}^n v(i) \\
 &\text{subject to } v(i) \leq v(j) \\
 &\qquad\qquad\qquad \text{if } i \in N \quad \text{and} \quad (i, j) \in E \\
 &\qquad\qquad\qquad v(i) = \frac{1}{2}[v(j) + v(k)] \\
 &\qquad\qquad\qquad \text{if } i \in A \quad \text{and} \quad (i, j), (i, k) \in E \\
 &\qquad\qquad\qquad v(n - 1) = 0 \\
 &\qquad\qquad\qquad v(n) = 1.
 \end{aligned}
 \tag{1}$$

Observe that the condition that  $H$  halts with probability 1 ensures the boundedness of (1): for each  $i \in V$ ,  $v(i) \leq 1$ .

Also observe that the size of the linear program (1) is polynomial in  $n$ , and therefore can be solved in time polynomial in  $n$  using an algorithm such as that of Khachiyan (1979). (See Schrijver (1986) for an exposition of polynomial-time algorithms for linear programming.) Thus it follows directly from Lemma 1 that for any given simple stochastic game  $G$  that halts with probability 1 and any

given strategy  $\sigma$  for player 1, a strategy for player 0 that is optimal with respect to  $\sigma$  can be found in polynomial time by solving a linear program.

**DEFINITION 6.** Let  $\sigma, \tau$  be a pair of strategies for players 1 and 0. These strategies are said to be *optimal* if each strategy is optimal with respect to the other.

Proofs of the remaining lemmas in this section can be found in (Condon, 1992a). (Most of these are based on proofs by Shapley (1953), Howard (1960), and Derman (1972).)

**LEMMA 2.** Let  $G$  be a simple stochastic game that halts with probability 1. Then there is a pair of optimal strategies  $\sigma^*, \tau^*$  for players 1 and 0 for the game  $G$ .

**DEFINITION 7.** The value of a simple stochastic game  $G$  is the value of the start vertex with respect to a pair of optimal strategies for the two players.

**LEMMA 3.** Given a simple stochastic game  $G$ , we can construct a new game  $G'$  in time polynomial in the size of  $G$  such that  $G'$  has the same number of min and max vertices as  $G$ , the value of  $G'$  is greater than  $\frac{1}{2}$  if and only if the value of  $G$  is greater than  $\frac{1}{2}$ , and  $G'$  halts with probability 1.

The next two lemmas show that our definition of “optimal” is suitable in the sense that an optimal strategy for either player optimizes the value of every vertex from that player’s point of view.

**LEMMA 4.** Let  $G = (V, E)$  be a simple stochastic game that halts with probability 1, and let  $\sigma^*, \tau^*$  be a pair of optimal strategies. Then for all  $i \in V$ ,

$$v_{\sigma^*, \tau^*}(i) = \max_{\sigma} \min_{\tau} v_{\sigma, \tau}(i).$$

**LEMMA 5.** Let  $G = (V, E)$  be a simple stochastic game that halts with probability 1. Then for any vertex  $i \in V$ ,

$$\min_{\tau} \max_{\sigma} v_{\sigma, \tau}(i) = \max_{\sigma} \min_{\tau} v_{\sigma, \tau}(i).$$

The following lemma is also necessary for showing the correctness of our algorithm.

**LEMMA 6.** Let  $G = (V, E)$  be a simple stochastic game that halts with probability 1, and let  $\sigma$  be a strategy for player 1 that is not optimal. Let  $i \in X$  be a vertex that is unstable with respect to  $\sigma, \tau(\sigma)$ . Let  $\sigma'$  be the strategy that is obtained from  $\sigma$  by changing the strategy at vertex  $i$ . Then for all  $j \in V$ ,  $v_{\sigma', \tau(\sigma')}(j) \geq v_{\sigma, \tau(\sigma)}(j)$ , and for some  $j \in V$ ,  $v_{\sigma', \tau(\sigma')}(j) > v_{\sigma, \tau(\sigma)}(j)$ .

### 3. THE ALGORITHM

*Input.* A simple stochastic game  $G = (V, E)$  that halts with probability 1, and some strategy  $\sigma$  for player 1.

*Output.* A pair of optimal strategies  $\sigma^*, \tau^*$  for players 1 and 0, respectively.

1. Choose uniformly at random a vertex  $s \in X$ .
2. Construct a new game  $\tilde{G} = (\tilde{V}, \tilde{E})$  as follows: Set  $\tilde{V} := V - \{s\}$ . Set  $\tilde{E} := (E - \{(j, k) \in E \mid j = s \text{ or } k = s\}) \cup \{(j, k) \mid (j, s) \in E \text{ and } l(s, k) = \sigma_s\}$ .
3. Recursively apply the algorithm to the game  $\tilde{G}$  and the player 1 strategy  $\tilde{\sigma} = (\sigma_1, \dots, \sigma_{s-1}, \sigma_{s+1}, \dots, \sigma_d)$  to find an optimal strategy  $\tilde{\sigma}'$  for player 1 for the game  $\tilde{G}$ . Extend  $\tilde{\sigma}'$  to a strategy  $\sigma'$  for  $G$  by setting  $\sigma'_s := \sigma_s$ .
4. Find an optimal strategy  $\tau'$  for player 0 with respect to the strategy  $\sigma'$ . If the pair  $\sigma', \tau'$  is optimal, then return  $\sigma', \tau'$ . Otherwise, set  $\sigma_k := \sigma'_k$  for  $k \neq s$  and  $\sigma_s := 1 - \sigma'_s$  and go back to step 1.

Before proceeding with the proof of the algorithm’s correctness, we introduce the following definitions.

**DEFINITION 8.**

- Define the function  $h(\sigma) = \sum_{i=1}^n v_{\sigma, \tau(\sigma)}(i)$ .
- We define a *switch* to be changing the strategy of player 1 at a single max vertex.
- We say that a switch is *profitable* if it moves from a strategy  $\sigma$  for player 1 to a strategy  $\sigma'$  with  $h(\sigma') > h(\sigma)$ .

**LEMMA 7.** The algorithm above finds a pair of optimal strategies, and every switch it makes is profitable.

*Proof.* We proceed by induction on  $d$ . If  $d = 1$ , then at most one switch is required—moving from the strategy that is not optimal to the strategy that is.

Now suppose that  $d > 1$ . Suppose that the current strategy at the beginning of a top-level iteration of the algorithm is  $\sigma$ . By the induction hypothesis, after choosing  $s, \sigma'$  is reached by some sequence of profitable switches, and it is the best strategy for player 1 that agrees with  $\sigma$  at vertex  $s$ . Therefore, every vertex other than  $s$  is stable, and if  $\sigma'$  is not optimal, then the vertex  $s$  is unstable. Therefore, by Lemma 6, changing the strategy at  $s$  is a profitable switch, and so every switch the algorithm makes is profitable. It follows that no strategy can be repeated in the sequence of switches made by the algorithm. Since there are only  $2^d$  possible strategies for player 1, the algorithm will terminate after no more than  $2^d - 1$  switches. Also observe that one switch is made at the end of each top-level iteration, regardless of the random choices, until a pair of optimal strategies is found. ■

### 4. ANALYSIS

**THEOREM 1.** The expected number of operations required by the algorithm is  $2^{O(\sqrt{\min\{|X|, |N|\}})} \times \text{poly}(n)$ .

*Proof.* At the bottom level of recursion, the strategy  $\tau'$  for player 0 in step 4 is found by solving a linear program

of size polynomial in  $n$ , as given by Lemma 1. At higher levels,  $\tau'$  can easily be constructed from the player 0 strategy returned by the recursive call in step 3. Therefore, only one such linear program is solved for each switch the algorithm makes. Thus, the total number of operations required per switch is polynomial in  $n$ , and the result follows from Lemmas 8 and 9 below. ■

Note that the first term in the expected operation count does not depend on the number of average vertices, but only on the number of min and max vertices. Therefore, transforming a given simple stochastic game into a game that halts with probability 1 per Lemma 3 does not affect this term.

LEMMA 8. *Let  $f(d)$  denote the expected number of switches required for the algorithm to find a pair of optimal strategies. Then*

$$f(d) \leq f(d-1) + \frac{1}{d} \sum_{i=1}^{d-1} f(i) + 1$$

for  $d > 1$  and

$$f(1) \leq 1.$$

*Proof.* Let  $\sigma^0$  be the initial strategy for player 1. Let  $h^i = \max_{\sigma} \{h(\sigma) \mid \sigma_i = \sigma_i^0\}$ . Let  $\{i_1, \dots, i_d\}$  be a permutation of  $\{1, \dots, d\}$  such that  $h^{i_1} \geq h^{i_2} \geq \dots \geq h^{i_d}$ . Now suppose that at step 1 the algorithm chooses at random vertex  $i_r$ . Then by solving a subproblem with  $d-1$  max vertices, it will reach a strategy  $\sigma'$  satisfying  $h(\sigma') = h^{i_r}$ . Then since every switch the algorithm makes is profitable, it can no longer make any switch to a strategy that agrees with  $\sigma^0$  at vertex  $i_j$ , for any  $j > r$ ; the strategy at each vertex  $i_j$  with  $j > r$  is now fixed until the algorithm terminates. By the same argument, the strategy at  $i_r$  will also remain fixed after one switch is made to correct it, if necessary. Therefore, after one top-level iteration requiring an expected number of switches not exceeding  $f(d-1) + 1$ , the size of the problem that remains to be solved is  $r-1$ . Then the recurrence follows from the fact that all possible choices of  $r$  are equally likely. ■

LEMMA 9.  *$f(d) \leq e^{2\sqrt{d-1}}$ , for all  $d \geq 1$ .*

*Proof.* Let  $g(d) = e^{2\sqrt{d-1}}$ . We proceed by induction on  $d$  to show  $g(d) \geq f(d)$ . First observe that  $g(1) = 1 \geq f(1)$ . Now we will show that  $f(d+1) \leq g(d+1)$  for  $d \geq 1$ . We have

$$\begin{aligned} f(d+1) &\leq f(d) + \frac{1}{d+1} \sum_{i=1}^d f(i) + 1 \\ &\leq g(d) + \frac{1}{d+1} \sum_{i=1}^d g(i) + 1 \end{aligned}$$

by the induction hypothesis. It remains to show that

$$g(d+1) \geq g(d) + \frac{1}{d+1} \sum_{i=1}^d g(i) + 1.$$

The function  $g$  is increasing monotonically, so

$$\begin{aligned} \sum_{i=1}^d g(i) &\leq \int_1^{d+1} g(x) dx \\ &= (d^{1/2} - 1/2) e^{2\sqrt{d}} + \frac{1}{2}. \end{aligned}$$

So we have

$$\begin{aligned} g(d) + \frac{1}{d+1} \sum_{i=1}^d g(i) + 1 &\leq e^{2\sqrt{d-1}} + \frac{d^{1/2}}{d+1} e^{2\sqrt{d}} - \frac{1}{2(d+1)} e^{2\sqrt{d}} \\ &\quad + \frac{1}{2(d+1)} + 1. \end{aligned} \tag{2}$$

The Taylor series expansion of  $g(d)$  about  $d+1$  gives

$$g(d) = g(d+1) - g'(d+1) + \frac{1}{2} g''(d+1) - \frac{1}{6} g'''(\xi)$$

for some  $\xi \in [d, d+1]$ . Observe that for  $x > 1$ ,

$$g'''(x) = \frac{4(x-1) - 6(x-1)^{1/2} + 3}{4(x-1)^{5/2}} e^{2\sqrt{x-1}} > 0.$$

Therefore,

$$\begin{aligned} g(d+1) &\geq g(d) + g'(d+1) - \frac{1}{2} g''(d+1) \\ &= e^{2\sqrt{d-1}} + \frac{1}{d^{1/2}} e^{2\sqrt{d}} - \frac{1}{2d} e^{2\sqrt{d}} + \frac{1}{4d^{3/2}} e^{2\sqrt{d}} \\ &\geq g(d) + \frac{1}{d+1} \sum_{i=1}^d g(i) + 1, \end{aligned}$$

where the final inequality follows from (2). ■

### 5. OPEN PROBLEMS

It remains to find a deterministic polynomial time algorithm for the simple stochastic game problem. Short of accomplishing that, an improved randomized algorithm or a deterministic subexponential algorithm would be a step in the right direction. Several algorithms are presented in (Condon, 1992b) without analysis. One such algorithm that seems to work well is due to Hoffman and Karp (1966): Start at some strategy  $\sigma$  for player 1. Find a strategy  $\tau(\sigma)$  for player 0 that is optimal with respect to  $\sigma$ . Find a new

strategy for player 1 by changing  $\sigma$  at every vertex that is unstable with respect to  $\sigma$ ,  $\tau(\sigma)$  and repeat.

Another approach is to consider the special case of linear programming when the graph corresponding to the feasible polyhedron is a  $d$ -dimensional cube. An improved simplex algorithm for this special case may lead to an improved algorithm for the simple stochastic game problem.

Also, the possibility of formulating the simple stochastic game problem directly as a linear program has not been ruled out. Condon (1992b) has shown that the problem can be formulated as a (non-convex) quadratic program that has an optimal solution at a vertex. Therefore, the same constraints coupled with an appropriate linear objective function would yield a linear program with the same optimal solution.

#### ACKNOWLEDGMENTS

I thank Anne Condon, who called to my attention the work done by Kalai (1992) on simplex algorithms for linear programming, and suggested applying these ideas to the simple stochastic game problem. I also thank Praseon Tiwari, who helped me to improve the presentation of the work in this paper.

Received November 20, 1992; final manuscript received July 19, 1993

#### REFERENCES

- Bland, R. G. (1977), New finite pivoting rules for the simplex method, *Math. Oper. Res.* **2**, 103–107.
- Chandra, A. K., Kozen, D. C., and Stockmeyer, L. J. (1981), Alternation, *J. Assoc. Comput. Mach.* **28**(1), 114–133.
- Condon, A. (1992a), The complexity of stochastic games, *Inform. and Comput.* **96**, 203–224.
- Condon, A. (1992), On algorithms for simple stochastic games, manuscript.
- Derman, C. (1972), "Finite State Markovian Decision Processes," Academic Press, New York.
- Filar, J. A. (1981), Ordered field property for stochastic games when the player who controls transitions changes from state to state, *J. Optimization Theory Appl.* **34**, 503–515.
- Gärtner, B. (1992), A subexponential algorithm for abstract optimization problems, in "33rd Annual Symposium on Foundations of Computer Science," pp. 464–472.
- Hoffman, A., and Karp, R. (1966), On nonterminating stochastic games, *Management Sci.* **12**, 359–370.
- Howard, R. A. (1960), "Dynamic Programming and Markov Processes," M.I.T. Press, Cambridge, MA.
- Kalai, G. (1992), A subexponential randomized simplex algorithm, in "Proceedings of the 24th Annual ACM Symposium on the Theory of Computing," pp. 475–482.
- Khachiyan, L. G. (1979), A polynomial algorithm in linear programming, *Soviet Math. Dokl.* **20**, 191–194.
- Matoušek, J., Sharir, M., and Welzl, E. (1992), A subexponential bound for linear programming, in "Proceedings of the 8th Annual Symposium on Computational Geometry."
- Melekopoglou, M., and Condon, A. (1990), "On the Complexity of the Policy Iteration Algorithm," Technical Report 941, University of Wisconsin–Madison Computer Sciences Department.
- Peters, H. J. M., and Vrieze, O. J. (1987), "Surveys in Game Theory and Related Topics," CWI Tract 39, Centrum voor Wiskunde en Informatica, Amsterdam.
- Raghavan, T. E. S., Tijs, S. H., and Vrieze, O. J. (1985), On stochastic games with additive reward and transition structure, *J. Optimization Theory Appl.* **47**, 451–464.
- Schrijver, A. (1986), "Theory of Linear and Integer Programming," Wiley–Interscience, New York.
- Shapley, L. S. (1953), Stochastic games, in "Proceedings of the National Academy of Sciences, U.S.A.," pp. 1095–1100.