On Algorithms for Simple Stochastic Games

Anne Condon[1]

Computer Sciences Department
University of Wisconsin-Madison
1210 West Dayton Street
Madison, WI 53706

Abstract

We survey a number of algorithms for the simple stochastic game problem, which is to determine the winning probability of a type of stochastic process, where the transitions are partially controlled by two players. We show that four natural approaches to solving the problem are incorrect, and present two new algorithms for the problem. The first reduces the problem to that of finding a locally optimal solution to a (non-convex) quadratic program with linear constraints. The second extends a technique of Shapley called the successive approximation technique, by using linear programming to maximize the improvement at each approximation step. Finally, we analyze a randomized variant of the Hoffman-Karp strategy improvement algorithm.

1   Introduction

In this paper, we study algorithms for the simple stochastic game problem. The problem is to find the winning probability of a type of stochastic process where transitions are partially controlled by two players. A simple stochastic game is a restriction of the general stochastic game introduced by Shapley [10] in 1953. Although this problem is known to be in $NP \cap$ co-NP (Condon [1]), no algorithm for the problem is known to run in polynomial time. We conjecture that the problem is in P and are interested in finding a polynomial time algorithm for the problem. Although we do not achieve this goal here, we present a number of new results on algorithms for the problem.

Our results are as follows. We present counterexamples that show that four natural approaches are incorrect. We also describe two new (correct) algorithms for the simple stochastic game problem. The first reduces the problem to that of finding a *locally optimal* solution to a (non-convex) quadratic program with linear constraints. The second extends a technique of Shapley [10], called the successive approximation technique, by using linear programming to maximize the improvement at each approximation step. Finally, we analyze a randomized variant of the Hoffman-Karp strategy improvement algorithm [5] and show that the number of iterations required is at most $2^{n-f(n)} + o(2^n)$, for any function $f(n) = o(n)$, where n is the number of nodes of the game.

The rest of this paper is organized as follows. In Section 1.1, we define the simple stochastic game problem and describe some fundamental properties of stochastic games. We describe previous work on algorithms for this problem in Section 1.3. In Section 2, we present four natural algorithms for the simple stochastic game problem and prove all are incorrect. Our new algorithms are described in Section 3. Conclusions and open problems are presented in Section 4.

1.1 Definition and Properties of Simple Stochastic Games

A *simple stochastic game (SSG)* is a directed graph $G = (V, E)$ with the following properties. The vertex set V is the union of disjoint sets $V_{max}, V_{min}, V_{average}$, called max, min and average vertices, together with two special vertices, called the *0-sink* and the *1-sink*. One vertex of V is called the *start* vertex. Each vertex of V has two outgoing edges, except the sink vertices, which have no outgoing edges. Without loss of generality, assume that the vertices of G are numbered $1, 2, \ldots n$, with $n - 1$ and n being the 0- and 1-sink vertices, respectively.

Associated with the game are two players, 0 and 1, who play the following game. Initially, a token is placed on the start vertex. The token is moved along edges of the graph, according to the following rules: whenever the token is on a min (max) vertex, player 0 (1) chooses the edge from that vertex along which the token is moved. Whenever the token is on an average vertex, the token is moved along an edge from that vertex which is chosen randomly and uniformly. The game ends when the token reaches a sink vertex. Player 1 wins if the token reaches the 1-sink vertex; otherwise player 0 wins.

A *min strategy* τ of player 0 is a set of edges of E, each with its left end at a min vertex such that for each min vertex i there is exactly one edge (i, j) in τ. Informally, if $(i, j) \in \tau$ then in a game where player 0 uses strategy τ, the token is always moved from vertex i to vertex j. Similarly, a max strategy σ of player 1 is a set of edges of E, each with its left end at a max vertex such that for each max vertex i there is exactly one edge (i, j) in σ.

Corresponding to strategy σ is a graph G_σ, which is the subgraph of G obtained by removing from each max vertex the outgoing edge that is not in the strategy σ. Similarly, corresponding to a pair of strategies σ and τ, is a graph $G_{\sigma,\tau}$ obtained from G_σ by removing from each min vertex the outgoing edge that is not in τ. In $G_{\sigma,\tau}$, every max and min vertex has one outgoing edge. Thus $G_{\sigma,\tau}$ can be considered as a Markov process where the states of the process are the vertices of G and the transition probabilities $q_{ij}, 1 \leq i, j \leq n$ are defined as follows. If $i \leq n-1$ then $q_{ij} = \frac{1}{2}$ if i is an average vertex with outgoing edge (i, j); $q_{ij} = 1$ if i is a max or min vertex with outgoing edge (i, j) and $q_{ij} = 0$ otherwise. Since $n - 1$ and n are sink states, we define $q_{nn} = q_{n-1n-1} = 1$, $q_{nj} = 0$ if $j \neq n$ and $q_{n-1j} = 0$ if $j \neq n - 1$.

We define the *value* $v_{\sigma,\tau}(i)$ of each vertex i of G with respect to strategies σ and τ to be the probability that player 1 wins the game if the start vertex is i and the players use strategies σ and τ. The *optimal value* of node i of G is defined to be $\max_\sigma \min_\tau v_{\sigma,\tau}(i)$. The *SSG problem* is: given a SSG G, is the optimal value of the start node of G at least $1/2$?

1.1.1 Stopping Games

We assume in this paper that an SSG is a *stopping game*, which means that for all pairs of strategies σ and τ, in the graph $G_{\sigma,\tau}$, there is a path from every node to a sink node. Condon [1] showed that the SSG problem is polynomial time many-one reducible to the SSG problem where instances are restricted to be stopping games. Hence, a polynomial time algorithm for stopping SSG's also yields a polynomial time algorithm for the unrestricted SSG problem.

Let σ and τ be strategies of a SSG G with n nodes. The values $v_{\sigma,\tau}(i)$ can be expressed as a solution to linear equations as follows. Let $\bar{v}_{\sigma,\tau} = (v_{\sigma,\tau}(1), \ldots, v_{\sigma,\tau}(n - 2))$. Let Q be the $(n - 2) \times (n - 2)$ matrix whose ijth entry is q_{ij}, that is, the probability of reaching j from i in one step. Let $\bar{b}$ be the $(n - 2)$-vector whose ith entry is q_{in}, that is, the probability of reaching the 1-sink from i in one step.

Lemma 1 $\bar{v}_{\sigma,\tau}$ *is the unique solution to the equation* $\bar{v}_{\sigma,\tau} = Q\bar{v}_{\sigma,\tau} + \bar{b}$. *Also,* $I - Q$ *is invertible, all entries of* $(I - Q)^{-1}$ *are non-negative and the entries along the diagonal are strictly positive.*

Uniqueness in the above lemma follows from the assumption that G is a stopping game. A proof can be found in [1].

1.1.2   Optimal Values and Strategies

We next summarize properties of the optimal values of a SSG. Shapley [10] (see also [1]) showed that for any node i,

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

We define the *optimal value vector* of G to be the vector of optimal values of the nodes of G. The optimal value vector of G is the unique solution to the following constraints (uniqueness follows from the assumption that G is a stopping game).

$$
\begin{aligned}
v(i) &= \max(v(j), v(k)), &&\text{if } i \text{ is a max node with children } j \text{ and } k\\
v(i) &= \min(v(j), v(k)), &&\text{if } i \text{ is a min node with children } j \text{ and } k\\
v(i) &= 1/2(v(j) + v(k)), &&\text{if } i \text{ is an average node with children } j \text{ and } k\\
v(n-1) &= 0\\
v(n) &= 1.
\end{aligned}
$$

We define max strategy σ to be *optimal with respect to* min strategy τ if for all max nodes i with child j, $v_{\sigma,\tau}(i) \geq v_{\sigma,\tau}(j)$. Similarly, τ is *optimal with respect to* σ if for all min nodes i with child j, $v_{\sigma,\tau}(i) \leq v_{\sigma,\tau}(j)$. Max strategy σ is *optimal* if for all i, $\min_{\tau} v_{\sigma,\tau}(i)$ equals the optimal value of i at G. Similarly, min strategy τ is optimal if for all i, $\max_{\sigma} v_{\sigma,\tau}(i)$ equals the optimal value of i at G.

Shapley showed that there is a pair of optimal strategies $\sigma(opt)$ and $\tau(opt)$ such that if $\bar{v}$ is the optimal value vector of the game, then for all $i, 1 \leq i \leq n$, $v(i) = v_{\sigma(opt),\tau(opt)}(i)$. The following lemma is proved in [1].

Lemma 2 *The optimal value of any node of a simple stochastic game with* n *vertices is of the form* p/q, *where* p *and* q *are integers,* $0 \leq p, q \leq 4^{n-1}$.

1.1.3   Generalized Games

In describing our algorithms and in proving them correct, it is sometimes convenient to generalize the definition of a SSG in two ways. First, we generalize the definition of a strategy. A *probabilistic* max (min) strategy is a weighting of the edges from the max (min) nodes, such that the weights are probabilities which sum to 1. The previous definition of a (pure) strategy is the special case where all weights are either 0 or 1. Second, we sometimes define a game to have more than two sink nodes. Each sink node has no outgoing edges and has an associated value in the range $[0,1]$. If a sink node s has value v, then the probability that player 1 wins, given that sink node s is reached, is v. Note that the 0-sink and 1-sink have values 0 and 1, respectively. Unless explicitly stated, our model of stochastic game is *not* generalized in these ways; when it is, we refer to the model as a generalized SSG.

The following lemma is used in Section 3 and is true even for generalized games.

Lemma 3 *Let G be a generalized SSG such that the sink nodes are numbered from j to n, with values $v(j), \ldots, v(n)$. Let σ and τ be max and min strategies of G. Then the value $v_{\sigma,\tau}(i)$ of node i is $\sum_{k=j}^{n} p_i(k)v(k)$, where $p_i(k)$ is the probability of reaching the kth sink from node i in $G_{\sigma,\tau}$.*

1.2 Notation

We use the following notation throughout the paper in describing our algorithms. Let G be a SSG with n nodes.

Let $\bar{v} = (v(1), \ldots, v(n))$, where $v(n-1) = 0$ and $v(n) = 1$. Let i be a node of G with children j and k. We say a node i of G is $\bar{v}$-*feasible* if the following is true. If i is an average node, $v(i) = 1/2(v(j) + v(k))$; if i is a max node, then $v(i) \geq \max\{v(j), v(k)\}$ and if i is a min node, then $v(i) \leq \min\{v(j), v(k)\}$. We say $\bar{v}$ is a *feasible vector* of G for all i, $1 \leq i < n-1$, i is $\bar{v}$-feasible.

We say node i is $\bar{v}$-*stable* if i is $\bar{v}$-feasible and furthermore, the following holds. If i is a max node, then $v(i) = \max\{v(j), v(k)\}$, and if i is a min node, then $v(i) = \min\{v(j), v(k)\}$. If i is not $\bar{v}$-stable, then i is $\bar{v}$-unstable. The vector $\bar{v}$ is *stable* if for all i, $1 \leq i < n-1$, i is $\bar{v}$-stable. Otherwise the vector is unstable. Note that from the results cited in Section 1.1.2, a stable vector is the unique optimal value vector of the game.

We say i is $\bar{v}$-*switchable* if i is a max node and $v(i) < \max\{v(j), v(k)\}$, or i is a min node and $v(i) > \min\{v(j), v(k)\}$. If σ (τ) is a strategy containing edge (i,j), we say σ' (τ') is obtained from σ (τ) by *switching* node i if $\sigma' = \sigma - \{(i,j)\} + \{(i,k)\}$ ($\tau' = \tau - \{(i,j)\} + \{(i,k)\}$). This can be generalized to switching sets of nodes.

1.3 Previous Work

Shapley [10] introduced the stochastic game model in 1953. Since then, numerous variations of Shapley's model have been studied, and many algorithms have been proposed. We have chosen to study the simple stochastic game because it is the simplest possible restriction of Shapley's model, which retains just enough complexity so that no polynomial time algorithm is known. Naturally, algorithms for the more general models also solve the SSG problem. In this section, we summarize three techniques that have been developed in previous work on general stochastic games, and apply them to our simple model.

The first is called successive approximation, introduced by Shapley [10], where a solution to the problem is found from an initial feasible vector, by repeatedly updating the value of each node based on the values of its children. This algorithm converges to the optimal value vector in the limit; however it can require exponential time to even get within a constant factor of this value. See the example of Figure 1. In this and all figures, min, max and average nodes are labeled N, X and A, respectively and the 0-sink and 1-sink nodes are labeled 0 and 1, respectively.

algorithm Successive Approximation

    let $\bar{v}$ be the feasible vector in which all min nodes have value 0 and all max
       nodes have value 1

    repeat

       let $\bar{v}'$ be defined as follows:

$$
\begin{aligned}
&v'(i) = \max\{v(j), v(k)\}, && \text{if } i \text{ is a max node with children } j \text{ and } k\\
&v'(i) = \min\{v(j), v(k)\}, && \text{if } i \text{ is a min node with children } j \text{ and } k\\
&v'(i) = 1/2(v(j) + v(k)), && \text{if } i \text{ is an average node with children } j \text{ and } k\\
&v'(n - 1) = 0\\
&v'(n) = 1
\end{aligned}
$$

       let $\bar{v} \leftarrow \bar{v}'$
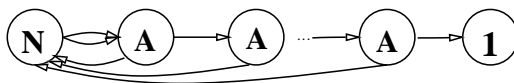
    until $\bar{v}$ is stable

Figure 1: Exponential time example for Shapley's Successive Approximation Algorithm

Suppose there are n average nodes in this figure. Then after the first iteration of the successive approximation algorithm, the value of the min node is $1/2^n$. After i iterations, the value is $1 - (1 - 1/2^n)^i$. Hence in the limit, as i goes to infinity, this is 1, but the rate of convergence is exponentially slow in n. Note that this example illustrates another useful fact: a feasible vector in which the difference between the value of all unstable nodes and their children is very small, may not be at all close to the optimal value vector. In this example, the initial vector is such that the only unstable node is the min node, and the difference between it and its child is only $1/2^n$. Yet, in the optimal value vector, the value of this min node is 1.

Another approach to solving general stochastic game problems is by reduction to mathematical programming problems, usually quadratic programming problems with non-convex objective functions, which are NP-complete. See Schultz [11] or Filar and Schultz [4] for a survey of such algorithms. When a SSG game is restricted to just two types of nodes – either max and average, or min and average – the problem is polynomial time solvable by reduction to linear programming. The proof is due to Derman [2]. He showed that if $G = (V, E)$ is a SSG with n vertices, none of which are min vertices, then the optimal value vector of G is the optimal solution to the following linear programming problem.

algorithm Linear Programming (for SSG's with no min vertices)

    minimize $\sum_{l=1}^{n} v(l)$, subject to the constraints

$$
\begin{aligned}
&v(i) \geq v(j), && \text{if } i \text{ is a max vertex and } (i, j) \in E\\
&v(i) = 1/2(v(j) + v(k)), && \text{if } i \text{ is an average vertex and } (i, j), (i, k) \in E\\
&v(i) \geq 0, && 1 \leq i \leq n\\
&v(n - 1) = 0\\
&v(n) = 1
\end{aligned}
$$

Note that the constraints simply ensure that the solution is feasible. Khachiyan [6] has shown that the linear programming problem is computable in time polynomial in the length of the input, which is

polynomial in n in this case. The proof for case (2), where G has just min and average vertices, is very similar.

A third method is known as the strategy improvement method, which was developed first for Markov decision processes, which are simple stochastic games with just max and average nodes, and no min nodes. This method was first proposed by Hoffman and Karp [5] for stochastic games. A pair of strategies of the players is chosen initially and repeatedly improved until an optimal strategy is found.

> **algorithm** Hoffman-Karp
>> let σ and τ be arbitrary max and min strategies, respectively
>> **repeat**
>>> let σ' be obtained from σ by switching all $\bar{v}_{\sigma,\tau}$-switchable max nodes
>>> let τ' be the min strategy that is optimal with respect to σ'
>>> let $\sigma \leftarrow \sigma'$, $\tau \leftarrow \tau'$
>> **until** $\bar{v}_{\sigma,\tau}$ is stable

The proof of correctness of this algorithm is very similar to our proof in Section 3 that a randomized variant is correct. The proof is based on the fact that at each iteration, for all i, $v_{\sigma',\tau'}(i) \geq v_{\sigma,\tau}(i)$ and if i is a $\bar{v}_{\sigma,\tau}$-switchable max node, the inequality is strict. Each iteration can be executed in polynomial time, by using Derman's linear programming algorithm to find τ' when σ' is fixed. It is not known if this algorithm requires an exponential number of iterations in the worst case. However, Melekopoglou and Condon [7] showed that many variations of the Hoffman-Karp algorithm require exponential time. In these variations, instead of switching all switchable max nodes in obtaining σ', only one switchable max node is switched.

Other relevant surveys of algorithms for stochastic games can be found in Vrieze [15] and Peters and Vrieze [8]. Also Van der Wal [13] and Federgruen [3] contain results on convergence rates for successive approximation schemes. Van der Wal [14] describes a variation of the strategy improvement algorithm of Hoffman and Karp in which an approximation to the optimal strategy is found at every step.

2 Incorrect Algorithms

In this section, we present four algorithms for the SSG problem. These algorithms use very different techniques, and seem to be natural algorithms for the problem. We show that all are incorrect.

2.1 Naive Linear Programming Algorithm

This algorithm extends the linear programming algorithm of Derman, described in the previous section, to simple stochastic games. The challenge is in designing a linear objective function which simultaneously minimizes the values of the max nodes and maximizes the values of the min nodes, subject to the constraint that the vector of values is feasible. In this algorithm, the objective function minimizes the sum of the values of the max nodes *minus* the sum of the values of the min nodes, subject to the constraints that the solution is feasible.

algorithm Naive Linear Programming

minimize $\sum_{l \in V_{max}} v(l) - \sum_{l \in V_{min}} v(l)$, subject to the constraints

$$
\begin{array}{ll}
v(i) \geq v(j), & \text{if } i \text{ is a max vertex and } (i,j) \in E \\
v(i) \leq v(j), & \text{if } i \text{ is a min vertex and } (i,j) \in E \\
v(i) = 1/2(v(j) + v(k)), & \text{if } i \text{ is an average vertex and } (i,j), (i,k) \in E \\
v(i) \geq 0, & 1 \leq i \leq n \\
v(n-1) = 0 & \\
v(n) = 1 &
\end{array}
$$

This algorithm is incorrect, as illustrated by the example of Figure 2. In the example, the value of all max and min nodes should be 1, in which case the value of the objective function is 1. However, by setting all the max and min nodes to 0, the constraints are still satisfied and the value of the objective function is 0.
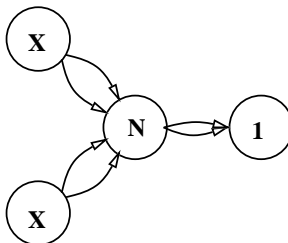

Figure 2: Counterexample to the Naive Linear Programming Algorithm

2.2   Modified Hoffman-Karp Algorithm

This algorithm is based loosely on the Hoffman-Karp algorithm. It proceeds in iterations, starting with an arbitrary pair of strategies σ and τ. It tries to improve on this pair by finding a max strategy σ' which is optimal with respect to τ, and a min strategy τ' which is optimal with respect to σ'. Thus it differs from the Hoffman-Karp algorithm by finding an *optimal* σ', instead of just switching the $\bar{v}_{\sigma,\tau}$-switchable max nodes to obtain σ'. Linear programming can be used to find the strategies in each iteration, using Derman's technique, so that each iteration can be completed in polynomial time.

algorithm Modified Hoffman-Karp

let σ and τ be arbitrary max and min strategies, respectively
repeat
    let σ' be an optimal max strategy with respect to τ
    let τ' be an optimal min strategy with respect to σ'
    let $\sigma \leftarrow \sigma'$, $\tau \leftarrow \tau'$
until $\bar{v}_{\sigma,\tau}$ is stable

This algorithm is incorrect, as illustrated in Figure 3. In this example, there are two max nodes, numbered 1 and 2, and two min nodes, numbered 3 and 4. In addition, for convenience in specifying strategies, two average nodes are numbered 5 and 6 and three sink nodes are numbered 7, 8 and 9. These numbers appear in parentheses as labels of the corresponding nodes. We use generalized sink nodes for simplicity, but these can be replaced by average nodes which are further connected to the 0- and 1-sink

nodes. The sink nodes are labeled by their values, for example, sink nodes numbered 7 and 8 have values .9 and .4, respectively.

Suppose that the initial strategies are $\sigma = \{\{1,3\},\{2,6\}\}$ and $\tau = \{\{3,8\},\{4,1\}\}$. In the first iteration, the new σ is obtained by switching all edges of the initial strategy σ, and subsequently, the new τ is obtained by switching all edges of the initial τ. In the second iteration, the new strategies are obtained by switching all edges yet again. Thus, the third iteration is the same as the first, the fourth iteration is the same as the second, and so on. The algorithm never finds the optimal strategies, which are $\sigma = \{\{1,7\},\{2,6\}\}$ and $\tau = \{3,8\},\{4,9\}\}$. Table 1 lists the values of the nodes as the algorithm proceeds through the first two iterations.
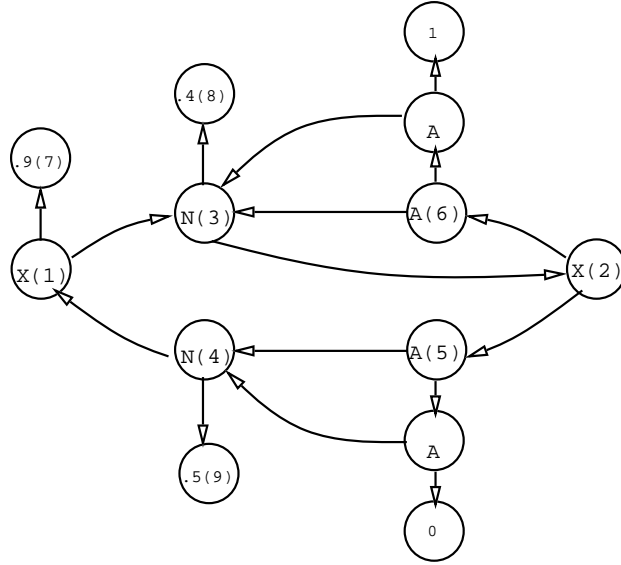

Figure 3: Counterexample to the Modified Hoffman-Karp Algorithm

Iteration		Strategies		Values of Nodes					
		σ	τ	1	2	3	4	5	6
0		$\{\{1,3\},\{2,6\}\}$	$\{\{3,8\},\{4,1\}\}$	0.4	0.55	0.4	0.4	0.3	0.55
1	(obtain σ)	$\{\{1,7\},\{2,5\}\}$	$\{\{3,8\},\{4,1\}\}$	0.9	0.675	0.4	0.9	0.675	0.55
	(obtain τ)	$\{\{1,7\},\{2,5\}\}$	$\{\{3,2\},\{4,9\}\}$	0.9	0.375	0.375	0.5	0.375	0.53125
2	(obtain σ)	$\{\{1,3\},\{2,6\}\}$	$\{\{3,2\},\{4,9\}\}$	1.0	1.0	1.0	0.5	0.375	1.0
	(obtain τ)	$\{\{1,3\},\{2,6\}\}$	$\{\{3,8\},\{4,1\}\}$	0.4	0.55	0.4	0.4	0.3	0.55

Table 1: Values of nodes of SSG of Figure 3 in the first two iterations of the Modified Hoffman-Karp Algorithm

2.3 Pollatschek Avi-Itzhak Algorithm

This algorithm was proposed by Pollatschek and Avi-Itzhak [9] for general stochastic games. It is similar to the Hoffman-Karp algorithm, except that instead of finding optimal strategies at each iteration, it simply modifies both strategies from the previous iteration by switching all switchable nodes. Pollatschek and Avi-Itzhak proved that the method is correct under very strong assumptions. Rao, Chandrasekaran

and Nair [12] gave a proof that the algorithm is correct for a general class of stochastic games. Van Der Wal [14] pointed out that their proof is incorrect and gave a counterexample. His example is a general stochastic game, not a SSG. We next describe this algorithm and show that on the example of Figure 3, the algorithm is incorrect.

algorithm Pollatschek Avi-Itzhak

    let σ and τ be arbitrary max and min strategies, respectively

    repeat

        let σ' be obtained from σ by switching all $\bar{v}_{\sigma,\tau}$-switchable max nodes

        let τ' be obtained from τ by switching all $\bar{v}_{\sigma,\tau}$-switchable min nodes

        let $\sigma \leftarrow \sigma'$, $\tau \leftarrow \tau'$

    until $\bar{v}_{\sigma,\tau}$ is stable

Again, consider the example of Figure 3. Table 2 illustrates the progress of the Pollatschek Avi-Itzhak algorithm, when the initial strategies are $\sigma = \{\{1,7\},\{2,6\}\}$ and $\tau = \{\{3,8\},\{4,9\}\}$. After four iterations, the algorithm cycles.

Iteration	Strategies		Values of Nodes					
	σ	τ	1	2	3	4	5	6
0	$\{\{1,7\},\{2,6\}\}$	$\{\{3,2\},\{4,9\}\}$	0.9	1.0	1.0	0.5	0.375	1.0
1	$\{\{1,3\},\{2,6\}\}$	$\{\{3,8\},\{4,9\}\}$	0.4	0.55	0.4	0.5	0.375	0.55
2	$\{\{1,7\},\{2,6\}\}$	$\{\{3,8\},\{4,1\}\}$	0.9	0.55	0.4	0.9	0.675	0.55
3	$\{\{1,7\},\{2,5\}\}$	$\{\{3,8\},\{4,9\}\}$	0.9	0.375	0.4	0.5	0.375	0.55
4	$\{\{1,7\},\{2,6\}\}$	$\{\{3,2\},\{4,9\}\}$	0.9	1.0	1.0	0.5	0.375	1.0

Table 2: Values of nodes of SSG of Figure 3 in the first four iterations of the Pollatschek Avi-Itzhak algorithm

2.4   The Naive Converge From Below Algorithm

This algorithm is based on the successive approximation algorithm of Shapley. To motivate this algorithm, we first describe a correct algorithm that requires exponential time.

This correct algorithm, the modified successive approximation algorithm, differs from Shapley's successive approximation algorithm as follows. The sequence of vectors are such that all max nodes are stable, so only the min nodes are unstable. In the initial vector $\bar{v}$, the values of all min nodes are 0 and the max nodes are $\bar{v}$-stable. (To see that such a vector exists, consider the generalized game G' obtained from the original game G, by removing all outgoing edges from the min nodes and changing them to sink nodes with value 0. Then, by Lemma 3, if $\bar{v}$ is the optimal value vector of the modified game G', then all max nodes of G' are $\bar{v}$-stable. This vector $\bar{v}$ is clearly a feasible vector of the original game G, in which all min nodes have value 0 and all max nodes are $\bar{v}$-stable). At each iteration, a new vector $\bar{v}'$ is found in which the values of the min nodes are increased just as in Shapley's algorithm, and the max nodes remain stable. This vector can be found in polynomial time using linear programming, in a way similar to Derman's algorithm of Section 1.3.

algorithm Modified Successive Approximation

    let $\bar{v}$ be the feasible vector in which all min nodes have value 0 and all max
       nodes are $\bar{v}$-stable

    repeat

       let $\bar{v}'$ be the feasible vector such that for all min nodes i with children j
         and k, $v'(i) = \min\{v(j), v(k)\}$, and all max nodes are stable
       let $\bar{v} \leftarrow \bar{v}'$

    until $\bar{v}$ is stable

This algorithm converges in the limit to the correct solution. However, it also requires exponential time to come even within a constant factor of the limit, in the example of Figure 1.

The naive converge from below algorithm is presented next. This algorithm tries to make more progress than the last algorithm, by using linear programming to maximize the increases of the min nodes at every iteration. In order to do this, the max nodes must be constrained. The linear program constrains the value of each max node to equal the child which currently has the larger value and to be at least the value of the other child. If both children have the same value, it constrains the value of the max node to equal one child, and alternates which child at alternate iterations. The following algorithm makes this precise.

algorithm Naive Converge From Below

    let $\bar{v}$ be the feasible vector in which all min nodes have value 0 and all max
       nodes are $\bar{v}$-stable
    let σ be any max strategy such that if $(i, j) \in \sigma$, then $v(i) = v(j)$

    repeat

       let $\bar{x}$ be the optimal solution to the following linear program:
       maximize $\sum_{i=1}^{n} x(i)$ subject to the following constraints

$x(i) \leq x(j)$,	if i is a min node with child j
$x(i) = 1/2(x(j) + x(k))$,	if i is an average node with children j and k
$x(i) = x(j)$ and $x(i) \geq x(k)$,	if i is a max node with children j and k
	such that (i) $v(j) > v(k)$ or
	(ii) $v(j) = v(k)$ and $(i, j) \in \sigma$
$x(n-1) = 0$	
$x(n) = 1$	

       let σ' be obtained from σ by switching all max nodes i with children j
         and k where $x(j) = x(k)$
       let $\bar{v} \leftarrow \bar{x}$, $\sigma \leftarrow \sigma'$

    until $\bar{v}$ is stable

This algorithm is incorrect, as illustrated by Figure 4. In this example, min node 3 has value 0 initially, and the max nodes have value 1/2. Let σ initially be $\{\{1, 2\}, \{2, 4\}\}$. After the first iteration, the values of all max and min nodes are 1/2. Hence σ becomes $\{\{1, 4\}, \{2, 3\}\}$. Thus, in the second iteration, no value changes, since the value of node 2 is constrained to be at most the value of node 4, which is 1/2. After the second iteration, $\sigma = \{\{1, 2\}, \{2, 4\}\}$ again, so the third iteration is the same as the first. Thus, the values remain 1/2, though the optimal value for the max and min nodes is 1.
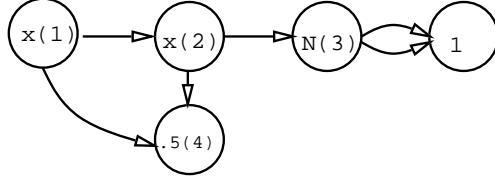
Figure 4: Counterexample to the Naive Converge from Below Algorithm

3 Correct Algorithms

In this section, we describe three correct algorithms for the SSG problem. The first reduces the problem to that of finding a *locally optimal* solution to a (non-convex) quadratic program with linear constraints. The second extends the successive approximation technique of Shapley, by using linear programming to maximize the improvement at each approximation step. Finally, we analyze a randomized variant of the Hoffman-Karp strategy improvement algorithm and show that the number of iterations required is at most $2^{n-f(n)} + o(2^n)$, for any function $f(n) = o(n)$.

3.1 A Quadratic Programming Algorithm

In this section, we describe a quadratic program with linear constraints whose solution is the optimal value vector of a SSG G with n nodes. Moreover, the optimal solution is the only *locally optimal* solution to the quadratic program in the constrained region.

The quadratic program is as follows: minimize

$$F(\bar{v}) = \sum_{\substack{i \in V_{max} \text{ or } V_{min} \\ \text{with children} j, k}} (v(i) - v(j))(v(i) - v(k))$$

subject to the constraints

$$
\begin{aligned}
&v(i) \geq v(j), && \text{if } i \text{ is a max node with child } j \\
&v(i) \leq v(j), && \text{if } i \text{ is a min node with child } j \\
&v(i) = 1/2(v(j) + v(k)), && \text{if } i \text{ is an average node with children } j \text{ and } k \\
&v(n-1) = 0 \\
&v(n) = 1
\end{aligned}
$$

The constraints guarantee that every term in the sum of the objective function is always non-negative in the constrained region. In fact it is clear that the objective function $F(\bar{v})$ is 0 if and only if the vector $\bar{v}$ is the optimal value vector of the game.

We claim that 0 is the only locally optimal solution of the objective function in the feasible region. Suppose to the contrary that $F(\bar{v})$ is another locally optimal solution to the problem in the feasible region. Let g be any number $\leq \frac{1}{2}\min\{|v(i) - v(j)| \mid i \text{ is a } \bar{v}\text{-unstable node with child } j\}$. Note that $g > 0$. We show how to find a feasible solution $\bar{v}'$ such that $|v(i) - v'(i)| \leq g$, for all i and $F(\bar{v}') < F(\bar{v})$. This proves that $\bar{v}$ cannot be a locally optimal solution.

The vector $\bar{v}'$ is defined as follows. If i is a $\bar{v}$-unstable min node, then $v'(i) = v'(i) + g$. Similarly, if i is a $\bar{v}$-unstable max node, then $v'(i) = v'(i) - g$. Furthermore, all other nodes are stable.

We need to show that (i) for all i, $|v'(i) - v(i)| \leq g$; (ii) $\bar{v}'$ lies in the constrained region of the quadratic program and (iii) $F(\bar{v}') < F(\bar{v})$.

To show (i), we show that for any i, $v'(i) \leq v(i) + g$; a similar argument shows that $v'(i) \geq v(i) - g$. It is clearly true for nodes i which are v-unstable nodes, by definition of $\bar{v}'$. Let G' and G'' be the games obtained from G by removing all edges from the $\bar{v}$-unstable nodes of G so that they become generalized sink nodes, and setting the value of such a node k to be $v'(k)$ and $v(k)$, respectively. Then $\bar{v}'$ and $\bar{v}$ are the optimal value vectors of the games G' and G'', respectively.

Let $\tau(opt)$ be an optimal min strategy of G''. Let $v'_{\sigma,\tau}(i)$ denote the value of i in game G' when the max and min strategies are σ and τ. We claim that in the game G', for an arbitrary strategy σ, for any i, $v'_{\sigma,\tau(opt)}(i) \leq v(i) + g$. If this claim is true, then

$$v'(i) = \min_{\tau} \max_{\sigma} v'_{\sigma,\tau}(i) \leq \max_{\sigma} v'_{\sigma,\tau(opt)}(i) \leq v(i) + g.$$

To prove the claim, fix i and let $p_i(k)$ be the probability of reaching the sink node k, when strategies $\sigma, \tau(opt)$ are used. (This probability is the same in G'' and G'). Then $v(i) \geq \sum_{k=j}^{n} p_i(k)v(k)$, since $v(i)$ is the optimal value of i in G'' and $\sum_{k=j}^{n} p_i(k)v(k)$ is the value of i when the strategies are σ and $\tau(opt)$. Hence,

$$v'_{\sigma,\tau(opt)}(i) = \sum_{k=j}^{n} p_i(k)v'(k) \leq \sum_{k=j}^{n} p_i(k)(v(k) + g) \leq v(i) + g.$$

We next show (ii), that the vector $\bar{v}'$ lies in the constrained region of the quadratic program. If i is a $\bar{v}$-stable node of G, then it is also a $\bar{v}'$-stable node and hence clearly satisfies the constraints of the program. Hence, suppose that i is a $\bar{v}$-unstable max node; the argument for unstable min nodes is similar. We need to show that $v'(i) \geq v'(j)$, where j is a child of i. By definition of g, $v(i) - v(j) \geq 2g$. We have already shown that $v'(j) \leq v(j) + g$ and by definition $v'(i) = v(i) - g$. Hence,

$$v'(i) - v'(j) \geq (v(i) - g) - (v(j) + g) = v(i) - v(j) - 2g \geq 0.$$

Finally, we show (iii), that $F(\bar{v}') < F(\bar{v})$. To see this, let i be a node with children j and k. If i is a $\bar{v}$-stable node, so that the term $(v(i) - v(j))(v(i) - v(k))$ is 0, then it is also the case that i is $\bar{v}'$-stable, and so $(v'(i) - v'(j))(v'(i) - v'(k))$ is 0. If i is $\bar{v}$-unstable, say a min node (the argument is similar for a max node), then

$$v'(j) - v'(i) \leq (v(j) + g) - (v(i) + g) = v(j) - v(i) \text{ and}$$
$$v'(k) - v'(i) \leq (v(k) + g) - (v(i) + g) = v(k) - v(i).$$

Hence $F(\bar{v}') \leq F(\bar{v})$. To prove that $F(\bar{v}')$ is strictly less than $F(\bar{v})$, we show that for some $\bar{v}$-unstable i, $(v'(i) - v'(j))(v'(i) - v'(k)) < (v(i) - v(j))(v(i) - v(k))$. where j and k are the children of i.

Suppose to the contrary that for all $\bar{v}$-unstable nodes i, $v'(i) - v'(j) = v(i) - v(j)$ and $v'(i) - v'(k) = v(i) - v(k)$, where j and k are the children of i. Fix any unstable node u, say a min node (the argument is similar if u is a max node).

Let N be the set of nodes defined inductively as follows: First, $u \in N$. If $m \in N$, then if m is an average node, both its children are in N; if m is a min node with children l and p such that $v(l) \leq v(p)$ then l is in N and else p is in N; and if m is a max node with children l and p such that $v'(l) \geq v'(p)$, then l is in N, and else p is in N. We show next that for all nodes l in N, $v'(l) - v(l) = g$. From this is it

immediate that N contains no sink nodes, since clearly, $v'(n-1) = v(n-1) = 0$ and $v'(n) = v(n) = 1$. This implies that there exist strategies σ and τ such that in the graph $G_{\sigma,\tau}$, there is no path from u to a sink node. This contradicts our assumption that G is a stopping game. Hence it must be the case that for some unstable node i, $v'(i) - v'(j) < v(i) - v(j)$ or $v'(i) - v'(k) < v(i) - v(k)$.

We now prove that for all nodes l in N, $v'(l) - v(l) = g$. We do this by induction on the distance of l from u. The base case is when $l = u$; since u is a $\bar{v}$-unstable min node, by definition of $\bar{v}'$, $v'(u) = v(u) + g$. Now suppose that l is of distance ≥ 1 from u and that l's parent is node m for which $v'(m) - v(m) = g$. Let m's other child be p. If m is a $\bar{v}$-unstable node, then by assumption, $v'(m) - v'(l) = v(m) - v(l)$. Combining this with $v'(m) - v(m) = g$, it follows that $v'(l) = v(l) + g$. Otherwise, suppose that m is a $\bar{v}$-stable node. If m is an average node, then $g = v'(m) - v(m) = 1/2(v'(l) - v(l)) + 1/2(v'(p) - v(p))$ But we have already shown that $v'(l) - v(l) \leq g$ and $v'(p) - v(p) \leq g$. Hence both must be $= g$. If m is a $\bar{v}$-stable max node, then m is also $\bar{v}'$-stable and so $v'(m) = v'(l)$. Also, $v(m) = \max\{v(l), v(p)\} \geq v(l)$. Hence $v'(l) = v(m) + g \geq v(l) + g$. Again, since also $v'(l) \leq v(l) + g$, it must be the case that $v'(l) = v(l) + g$. Finally, suppose m is a $\bar{v}$-stable min node, and hence also $\bar{v}'$-stable. By construction, $v(m) = v(l)$. Also, $v'(m) \leq v'(l)$. Hence $v'(l) \geq v'(m) = v(m) + g = v(l) + g$. Again, since also $v'(l) \leq v(l) + g$, it must be the case that $v'(l) = v(l) + g$.

This completes the proof that $F(\bar{v}') < F(\bar{v})$ and hence the proof that 0 is a locally optimal solution to the quadratic programming problem.

3.2   Converge from Below Algorithm

This algorithm is a modification to the Naive Converge From Below algorithm. That algorithm attempted to repeatedly improve a feasible vector by maximizing the increase of the unstable min nodes, while constraining the max nodes to remain stable, using linear programming. However, Figure 4 showed that the constraints on the max nodes cause the algorithm to cycle on a suboptimal solution. To avoid this problem, we relax the constraints on the max nodes, so that the solution of the linear program may no longer be a feasible vector of the game. Linear programming is used again to produce a new feasible vector for the next iteration.

algorithm Converge From Below

> let $\bar{v}$ be the feasible vector in which all min nodes have value 0 and all max
> nodes are stable
>
> **repeat**
>
> > let $\bar{x}$ be the solution to the following linear program, $LP(\bar{v})$:
> > maximize $\sum_{i=1}^{n} x(i)$ subject to the following constraints
> >
> > $$x(i) \leq x(j), \qquad\qquad\quad \text{if } i \text{ is a min node with child } j$$
> > $$x(i) = 1/2(x(j) + x(k)), \quad \text{if } i \text{ is an average node with children } j \text{ and } k$$
> > $$x(i) = x(j), \qquad\qquad\quad \text{if } i \text{ is a max node with children } j \text{ and } k$$
> > $$\text{such that } v(j) > v(k)$$
> > $$x(i) = 1/2(x(j) + x(k)), \quad \text{if } i \text{ is a max node with children } j \text{ and } k$$
> > $$\text{such that } v(j) = v(k)$$
> > $$x(n-1) = 0$$
> > $$x(n) = 1$$
> >
> > let $\bar{v}'$ be the feasible vector such that $v'(i) = x(i)$ if i is a min node and
> > such that all max nodes are stable
> > let $\bar{v} \leftarrow \bar{v}'$
>
> **until** $\bar{v}$ is stable

We claim that this algorithm converges to the optimal value vector of G. We prove this as follows. Let $\bar{x}$ be the optimal solution to $LP(\bar{v})$. We first show that $\bar{x}$ exists and is the optimal value vector of some game with n nodes. Then, we show that $\bar{v}' \geq \bar{x} \geq \bar{v}$ and that the inequality $\bar{x} \geq \bar{v}$ is strict at some node. Finally, we put these two facts together to prove the claim.

Note that the vector $\bar{v}$ is a feasible solution to $LP(\bar{v})$; hence $\bar{x}$ exists. Also, $\bar{x}$ is the optimal value vector of the game G_r obtained from G as follows. All max nodes of G become average nodes in G_r. For each such node i with children j and k, if $v(j) = v(k)$ then i's children in G_r are still j and k; however if $v(j) > v(k)$ then both children of i are j. The fact that $\bar{x}$ is the optimal value vector of G_r follows from the fact that $\bar{x}$ is the optimal solution to $LP(\bar{v})$, which is exactly the instance of linear programming obtained from G_r by applying the linear programming algorithm described in Section 1.3. (In this case, because the game has just min and average nodes, rather than just max and average nodes, the linear programming problem maximizes, rather than minimizes the objective function, and the inequalities are reversed).

We now show that $\bar{x} \geq \bar{v}$. Note that $\bar{x}$ can be obtained from $\bar{v}$ in the limit by applying the successive approximation algorithm of Section 1.3 to the game G_r defined in the last paragraph. Moreover, since the game has no max nodes, the feasible vector can never decrease at any iteration of the algorithm. Hence, $\bar{x} \geq \bar{v}$. It is also straightforward to show that $\bar{v}' \geq \bar{x}$.

We next show that at some node l, $x(l) > v(l)$. Let l be a $\bar{v}$-unstable min node of G. We now construct a feasible solution $\bar{x}$ of $LP(\bar{v})$, such that $x(l) > v(l)$. From this, it follows that the optimal solution to $LP(\bar{v})$ must be strictly greater than $\bar{v}$ at some node. Before defining $\bar{x}$, we need to define a restricted game G' in the next paragraph.

Let G' be the game obtained from G by removing all edges from every min node so that they become generalized sink nodes, and setting the value of such a node k to $v(k)$. Without loss of generality, assume that the sink nodes of G' are numbered $j \ldots n$. Let σ be the following generalized max strategy of G'. Let i be any max node with children m and p. If $v(m) > v(p)$ then the strategy at i is to go to m with

probability 1. Otherwise $v(m) = v(p)$, in which case the strategy is to go to each node with probability $1/2$. With respect to this strategy, let $p_i(k)$ be the probability that sink node k is reached from node i, $j \leq k \leq n$. Then from Lemma 3, $v(i) = \sum_{k=j}^{n} p_i(k)v(k)$.

Let $g > 0$ be such that for each child l' of the unstable min node l, $g \leq v(l') - v(l)$. We now define $\bar{x}$ as follows. For all min nodes $i \neq l$, let $x(i) = v(i)$ and let $x(l) = v(l) + g$. For each remaining node i, let $x(i) = \sum_{k=j}^{n} p_i(k)x(k)$. Equivalently, $x(i)$ is the value of node i in the restricted game G'' defined as follows, when the max strategy is σ: All edges from every min node are removed so that the min nodes become generalized sink nodes, and the value of such a node k is $x(k)$.

By construction, for $j \leq k \leq n$, $v(k) \leq x(k) \leq v(k) + g$. This can be extended to show that in fact for all i, $v(i) \leq x(i) \leq v(i) + g$, as follows:

$$v(i) = \sum_{k=j}^{n} p_i(k)v(k) \leq \sum_{k=j}^{n} p_i(k)x(k) = x(i) \leq \sum_{k=j}^{n} p_i(k)(v(k) + g) = v(i) + g.$$

We claim that $\bar{x}$ is a feasible solution to $LP(\bar{v})$. Value $x(l)$ is at most the value of its children since its value is $x(l) = v(l) + g \leq v(j) \leq x(j)$, where j is any child of l. For min node $i \neq l$, $x(i) = v(i) \leq v(j) \leq x(j)$, so again $x(i) \leq x(j)$. If i is an average node with children j and k then $x(i) = 1/2(x(j) + x(k))$ since the value $x(i)$ is the value of node i in G'' when the max strategy is σ. Finally consider a max node i with children j and k. Suppose first that $v(j) = v(k)$. Then we need to show that $x(i) = 1/2(x(j) + x(k))$. But strategy σ is defined so that this is true. Similarly, if $v(j) > v(k)$, then σ is defined so that $x(i) = x(j)$.

We have now shown the following. Suppose that $\bar{v}$, $\bar{v}'$ and $\bar{v}''$ are the feasible vectors at the start of three successive iterations. Let $\bar{x}$ and $\bar{x}'$ be the optimal solutions to $LP(\bar{v})$ and $LP(\bar{v}')$ respectively. Then, $\bar{v}'' \geq \bar{x}' \geq \bar{v}' \geq \bar{x} \geq \bar{v}$. Moreover, the inequality $\bar{x}' \geq \bar{x}$ is strict at some node, say i. Since $\bar{x}'$ and $\bar{x}$ are solutions to games with n nodes, by Lemma 2, $x'(i)$ and $x(i)$ are of the form p/q, $0 \leq p, q \leq 4^{n-1}$ and hence $x'(i) - x(i) \geq 1/4^{n-1}$. This rate of improvement, together with the fact that the vector $\bar{v}$ at the start of each iteration is bounded above by the optimal value vector of G, imply that in $2^{O(n)}$ iterations the optimal value vector will be found. $\square$

3.3   A Randomized Hoffman-Karp Algorithm

In this section, we analyze a randomized variant of the Hoffman-Karp algorithm.

algorithm Randomized Hoffman-Karp

choose arbitrary max and min strategies σ and τ respectively

repeat

randomly and uniformly, chose $2n$ non-empty subsets of the max nodes that are $\bar{v}_{\sigma,\tau}$-switchable (these subsets need not be distinct)

let the strategies obtained from σ by switching these subsets be $\sigma_1, \dots, \sigma_{2n}$

let $\tau_1, \dots \tau_{2n}$ be optimal strategies of the min player with respect to $\sigma_1, \dots, \sigma_{2n}$, respectively

let l be such that $\sum_{i=1}^{n} v_{\sigma_l,\tau_l}(i) \geq \sum_{i=1}^{n} v_{\sigma_m,\tau_m}(i)$, where $1 \leq l, m \leq 2n$

let $\sigma \leftarrow \sigma_l$, $\tau \leftarrow \tau_l$

until $\bar{v}_{\sigma,\tau}$ is stable

We first show that this algorithm is correct, following the proof of Hoffman and Karp. If the algorithm halts, the strategies σ and τ of the last iteration are clearly optimal, since $\bar{v}_{\sigma,\tau}$ is stable. Hence it is sufficient to prove that the algorithm halts.

We call the strategies σ and τ at the start of an iteration the *current* strategies of the iteration. Suppose that the current strategies of two successive iterations of the repeat loop, are σ, τ and σ', τ', respectively. Let $v(i)$ and $v'(i)$ be the value of node i, $1 \leq i \leq n$, with respect to strategies σ, τ and σ', τ', respectively. We show that for all i, $v(i) \leq v'(i)$ and the inequality is strict for some i. From this it follows that the algorithm halts, since there are only a finite number of strategies, and no pair can be repeated at the start of an iteration.

We now show that for all i, $v(i) \leq v'(i)$ and the inequality is strict for some i. Clearly, the values are equal at the sink nodes, which we assume are numbered $n-1$ and n. Let $\bar{v} = (v(1), \dots v(n-2))^T$, $\bar{v}' = (v(1), \dots v(n-2))^T$, $\bar{v} = Q\bar{v} + \bar{b}$ and $\bar{v}' = Q'\bar{v}' + \bar{b}'$ for some $Q, Q', \bar{b}$ and $\bar{b}'$, as in Lemma 1.

Let $\bar{\Delta} = \bar{v}' - \bar{v}$. We show that $\bar{\Delta} \geq 0$ and that some entry is actually > 0. Adding and subtracting $Q'\bar{v} + \bar{b}'$ to $\bar{\Delta}$ we see that

$$\bar{\Delta} = (Q'\bar{v}' + \bar{b}') - (Q'\bar{v} + \bar{b}') + (Q'\bar{v} + \bar{b}') - (Q\bar{v} + \bar{b}).$$

If $\bar{\delta} = (Q'\bar{v} + \bar{b}') - (Q\bar{v} + \bar{b})$, then $\bar{\Delta} = Q'\bar{\Delta} + \bar{\delta}$.

Again from Lemma 1, we know that the matrix $(I - Q')$ is invertible, since the game halts with probability 1. Hence $\bar{\Delta} = (I - Q')^{-1}\bar{\delta}$. Also, all the entries of $(I - Q')^{-1}$ are ≥ 0 and the entries along the diagonal are > 0. Therefore, it remains to show that $\bar{\delta} \geq 0$ and that one entry is actually > 0.

Note that Q and Q' differ only in rows i where the edge at node i has been switched in obtaining σ' and τ' from σ and τ. The same is true for $\bar{b}$ and $\bar{b}'$. If the ith row of Q and Q' are equal and also the ith row of $\bar{b}$ and $\bar{b}'$ are equal, then clearly the ith entry of δ is 0.

Otherwise, suppose that node i has children j and k and that the ith entries of $Q'\bar{v} + \bar{b}'$ and $Q\bar{v} + \bar{b}$ are $v(k)$ and $v(j)$, respectively. Thus, the ith entry of $\bar{\delta}$ is $v(k) - v(j)$. There are two cases, depending on whether i is a max or a min node.

First, suppose that i is a max node. Since edge (i,k) replaces (i,j) in constructing σ' from σ, it must be the case that $v(k) > v(j)$. Hence the ith entry of $\bar{\delta}$ must be > 0. Otherwise, i is a min node. Since τ is optimal with respect to σ and $(i,j) \in \tau$, it must be the case that $v(j) \leq v(k)$. Hence the ith entry of $\bar{\delta}$ is again ≥ 0.

We next analyze the number of iterations required by the algorithm. Let n be the number of max nodes. We claim that the expected number of iterations of this algorithm is $2^{n-f(n)} + 2^{o(n)}$, for any function $f(n) = o(n)$, where n is the number of max nodes.

We consider two types of iteration of the algorithm. We call an iteration *good* if there are at least $f(n) + 2$ $\bar{v}_{\sigma,\tau}$-switchable max nodes, where σ and τ are the current strategies of the iteration. Otherwise we call an iteration *bad*. We claim that there are at most $2^{o(n)}$ bad iterations. This is true since $f(n)$ is $o(n)$, and so the number of subsets of n of size $f(n)$ is $2^{o(n)}$.

We next consider good iterations. Associated with any iteration, there is a set of *candidate* strategies, which are all those strategies that could be the current strategy of a future iteration of the algorithm. A candidate strategy is *eliminated* in that iteration, if it is no longer a candidate strategy in the next iteration. Our goal is to show that if an iteration is good, then with probability $1 - 2^{-2n}$, at least

$2^{f(n)}$ of these candidate strategies are eliminated. From this it follows that with probability at least $(1 - 2^{-2n})^{2^n} \geq 1 - 2^{-n}$, there are at most $2^{n-f(n)}$ good iterations.

Let σ and τ be the current strategies associated with some good iteration and let σ' be the current max strategy of the next iteration. Let S be the set of strategies obtained by switching all the distinct non-empty subsets of the $\bar{v}_{\sigma,\tau}$-switchable nodes. Note that all the strategies of S are candidate strategies of this iteration, and since it is a good iteration, $|S| \geq 2^{f(n)+2} - 1$. Let the value of a max strategy σ_j be $\sum_{i=1}^{n} v_{\sigma_j, \tau_j}(i)$, where τ_j is the optimal min strategy with respect to σ_j. Order the strategies of S with respect to their values, and let S' be the set consisting of the best $\lceil |S|/2 \rceil$ strategies of S (ties are broken arbitrarily). Then the probability that $\sigma' \in S'$ is at least $1 - (1/2)^{2n}$. This is because σ' is the best of $2n$ randomly chosen strategies of S. Also, the probability that each of these randomly chosen strategies is in S' is $\geq 1/2$, since $'|S'| \geq |S|/2$.

If $\sigma' \in S'$, then all of the strategies in $S - S'$ are eliminated as candidate strategies of the next iteration. This is because all these strategies have value at most the value of σ'. Furthermore, we have already established that the candidate strategies of an iteration must have value strictly greater than the current strategy of that iteration. Also, $|S - S'| \geq (2^{f(n)+2} - 1) - 2^{f(n)+1} \geq 2^{f(n)}$. This completes the proof that at least $2^{f(n)}$ candidate strategies are eliminated at a good iteration, with probability at least $1 - 2^{-2n}$.

Therefore, expected number of iterations is at most

$$(1 - 2^{-n})2^{n-f(n)} + 2^{-n}2^n + 2^{o(n)} = 2^{n-f(n)} + 2^{o(n)}.$$

4 Conclusions

We have demonstrated that a number of natural approaches to solving the SSG problem are incorrect. We have also proposed two new deterministic algorithms and a randomized algorithm for the problem. A major open question is whether these algorithms run in polynomial time.

5 Acknowledgements

References

[1] Condon, A. The complexity of stochastic games, *Information and Computation*, 96(2):203-224, February 1992.

[2] Derman, C. *Finite State Markov Decision processes*, Academic Press, 1972.

[3] Federgruen, A. Successive approximation methods in undiscounted stochastic games, *Operations Research* 1:794-810, 1980.

[4] Filar, J. A. and T. Schultz, Non-linear programming and stationary strategies in stochastic games, *Mathematical Programming* 35:243-247, 1987.

[5] Hoffman, A. and Karp, R. On nonterminating stochastic games, *Management Science* 12:359-370, 1966.

[6] Khachiyan, L. G. A polynomial time algorithm for linear programming. *Soviet Math Dokl.*, 20:191-194, 1979.

[7] Melekopoglou, M. and A. Condon. On the Complexity of the Policy Iteration Algorithm for Simple Stochastic Games, University of Wisconsin-Madison Computer Sciences Department Technical Report Number 941, June 1990.

[8] Peters, H. J. M and O. J. Vrieze. Surveys in game theory and related topics, *CWI Tract 39*, Center for Mathematics and Computer Science, Amsterdam, 1987.

[9] Pollatschek, M. A. and B. Avi-Itzhak. Algorithms for stochastic games with geometrical interpretation, *Management Science* 15:399-415, 1969.

[10] Shapley, S. Stochastic games, *Proceedings of the National Academy of Sciences*, U.S.A, 39:1095-1100, 1953.

[11] Schultz, T. A. Mathematical Programming and Stochastic Games, Ph. D. Dissertation, Johns Hopkins University, Baltimore, Maryland, 1987.

[12] Rao, S. S., Chandrasekaran, R. and Nair, K. P. K. Algorithms for discounted stochastic games, *Journal of Optimization Theory and Applications*, 11(6):627-637,1973.

[13] Van Der Wal, J. Discounted Markov games: successive approximations and stopping times, *International Journal of Game Theory*, 6:11-22, 1977.

[14] Van Der Wal, J. Discounted Markov games: generalized policy iteration method, *Journal of Optimization Theory and Applications*, 25(1):125-138,1978.

[15] Vrieze, O.J., Stochastic games with finite state and action spaces, *CWI Tract 33*, Center for Mathematics and Computer Science, Amsterdam, 1987.