The complexity of solving reachability games using value and strategy iteration *

Kristoffer Arnsfelt Hansen, Rasmus Ibsen-Jensen, and Peter Bro Miltersen

Aarhus University
{arnsfelt,rij,pbmiltersen}@cs.au.dk

Abstract. Two standard algorithms for approximately solving two-player zero-sum concurrent reachability games are *value iteration* and *strategy iteration*. We prove upper and lower bounds of $2^{m^{\Theta(N)}}$ on the worst case number of iterations needed for both of these algorithms to provide non-trivial approximations to the value of a game with N non-terminal positions and m actions for each player in each position.

1 Introduction

1.1 Statement of problem and overview of results

We consider *finite state, two-player, zero-sum, deterministic, concurrent reachability games*. For brevity, we shall henceforth refer to these as just reachability games. The class of reachability games is a subclass of the class of games dubbed *recursive games* by Everett [8] and was introduced to the computer science community in a seminal paper by de Alfaro, Henzinger and Kupferman [6]. A reachability game G is played between two players, Player I and Player II. The game has a finite set of *non-terminal* positions and special *terminal* positions GOAL and TRAP. In this paper, we let N denote the number of non-terminal positions and assume positions are indexed $1, \ldots, N$ while GOAL is indexed $N + 1$ and TRAP 0. At any point in time during play, a *pebble* rests at some position. The position holding the pebble is called the *current* position. The objective for Player I is to eventually make the current position GOAL. If this happens, play ends and Player I wins. The objective for Player II is to *forever* prevent this from happening. This may be accomplished either by the pebble reaching TRAP from where it cannot escape or by it moving between non-terminal positions indefinitely. To each non-terminal position i is associated a finite set of actions A_i^1, A_i^2 for each of the two players. In this paper, we assume that all these sets have the same size m (if not, we may "copy" actions to make this so) and that $A_i^1 = A_i^2 = \{1, \ldots, m\}$. At each point in time, if the current position is i, Player I and Player II simultaneously choose actions in $\{1, \ldots, m\}$. For each position i and each action pair $(a, a') \in \{1, \ldots, m\}^2$ is

* Work supported by Center for Algorithmic Game Theory, funded by the Carlsberg Foundation. The authors acknowledge support from The Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, under which part of this work was performed.

associated a position $\pi(i, a, a')$. In other words, each position holds an $m \times m$ matrix of *pointers* to positions. When the current position at time t is i and the players play the action pair (a, a'), the new position of the pebble at time $t + 1$ is $\pi(i, a, a')$.

A *strategy* for a reachability game is a (possibly randomized) procedure for selecting which action to take, given the history of the play so far. A strategy *profile* is a pair of strategies, one for each player. A *stationary strategy* is the special case of a strategy where the choice only depends on the current position. Such a strategy is given by a family of probability distributions on actions, one distribution for each position, with the probability of an action according to such a distribution being called a *behavior probability*. We let $\mu_i(x, y)$ denote the probability that Player I eventually reaches GOAL if the players play using the strategy profile (x, y) and the pebble starts in position i. The *lower value* of position i is defined as: $\underline{v_i} = \sup_{x \in S^1} \inf_{y \in S^2} \mu_i(x, y)$ where S^1 (S^2) is the set of strategies for Player I (Player II). Similarly, the *upper value* of a position i is $\overline{v_i} = \inf_{y \in S^2} \sup_{x \in S^1} \mu_i(x, y)$. Everett [8] showed that for all positions i in a reachability game, the lower value $\underline{v_i}$ in fact equals the upper value $\overline{v}_i$, and this number is therefore simply called the *value* v_i of that position. The vector v is called the *value vector* of the game. Furthermore, Everett showed that for any $\epsilon > 0$, there is a stationary strategy x^* of Player I so that for all positions i, we have $\inf_{y \in S^2} \mu_i(x^*, y) \geq v_i - \epsilon$, i.e. the strategy x^* guarantees the value of any position within ϵ when play starts in that position. Such a strategy is called *ϵ-optimal*. Note that x^* does not depend on i. It may however depend on $\epsilon > 0$ and this dependence may be necessary, as shown by examples of Everett. In contrast, it is known that Player II has an exact optimal strategy that is guaranteed to achieve the value of the game, without any additive error [17, 13].

In this paper, we consider algorithms for *solving* reachability games. There are two notions of solving a reachability game relevant for this paper:

1. *Quantitatively*: Given a game, compute ϵ-approximations of the entries of its value vector (we consider approximations, rather than exact computations, as the value of a reachability game may be an irrational number).
2. *Strategically*: Given a game, compute an ϵ-optimal strategy for Player I.

Once a game has been solved strategically, it is straightforward to also solve it quantitatively (for the same ϵ) by analyzing, using linear programming, the finite state Markov decision process for Player II resulting when freezing the computed strategy for Player I. The converse direction is far from obvious, and it was in fact shown by Hansen, Koucký and Miltersen [12] that if standard binary representation of behavior probabilities is used, merely *exhibiting* an $(1/4)$-optimal strategy requires worst case exponential space in the size of the game. In contrast, a $(1/4)$-approximation to the value vector obviously only requires polynomial space to describe and it may be possible to compute it in polynomial time, though it is currently not known how to do so [5].

There is a large and growing literature on solving reachability games [6, 7, 3, 1, 2, 12]. In this paper, we focus on the two perhaps best-known and best-studied algorithms, *value iteration* and *strategy iteration*. Both were originally derived from similar algorithms for solving Markov decision processes [15] and *discounted* stochastic games [19]. We describe these algorithms next. Value iteration is Algorithm 1. Value iteration approximately solves reachability games quantitatively.

Algorithm 1: Value Iteration

```
1: t := 0
2: ṽ^0 := (0, ..., 0, 1) // the vector ṽ^0 is indexed 0, 1, ..., N, N + 1
3: while true do
4:     t := t + 1
5:     ṽ^t_0 := 0
6:     ṽ^t_{N+1} := 1
7:     for i ∈ {1, 2, ..., N} do
8:         ṽ^t_i := val(A_i(ṽ^{t−1}))
```

Algorithm 2: Strategy Iteration

```
1: t := 1
2: x^1 := the strategy for Player I playing uniformly at each position
3: while true do
4:     y^t := an optimal best reply by Player II to x^t
5:     for i ∈ {0, 1, 2, ..., N, N + 1} do
6:         v^t_i := μ_i(x^t, y^t)
7:     t := t + 1
8:     for i ∈ {1, 2, ..., N} do
9:         if val(A_i(v^{t−1})) > v^{t−1}_i then
10:            x^t_i := maximin(A_i(v^{t−1}))
11:        else
12:            x^t_i := x^{t−1}_i
```

In the pseudocode of Algorithm 1, the matrix $A_i(\tilde{v}^{t-1})$ denotes the result of replacing each pointer to a position j in the $m \times m$ matrix of pointers at position i with the real number $\tilde{v}_j^{t-1}$. That is, $A_i(\tilde{v}^{t-1})$ is a matrix of $m \times m$ real numbers. Also, $\text{val}(A_i(\tilde{v}^{t-1}))$ denotes the value of the *matrix game* with matrix $A_i(\tilde{v}^{t-1})$ and the row player being the maximizer. This value may be found using linear programming. Value iteration works by iteratively updating a *valuation* of the positions, i.e., the numbers $\tilde{v}_i^t$. Clearly, when implementing the algorithm, valuations $\tilde{v}_i^t$ only have to be kept for one iteration of the while loop after the iteration in which they are computed and the algorithm thus only needs to store $O(N)$ real numbers.[1] As stated, the algorithm is non-terminating, but has the property that as t approaches infinity, the valuations $\tilde{v}_i^t$ approach the correct values v_i from below. We present an easy (though not self-contained) proof of this well-known fact in section 2.1 below, and also explain the intuition behind the truth of this statement. However, until the present paper, there has been no published information on the number of iterations needed for the approximation to be an ϵ-approximation to the correct value for the general case of concurrent reachability games, though Condon [4] observed that for the case of *turn-based* games (or "simple stochastic games"), the number of iterations has to be at least exponential in N in order to achieve an ϵ-

[1] In this paper, we assume the real number model of computation and ignore the (severe) technical issues arising when implementing the algorithm using finite-precision arithmetic.

approximation. Clearly, the concurrent case is at least as bad. In fact, this paper will show that the concurrent case is much worse!

Strategy iteration is Algorithm 2. It approximately solves reachability games quantitatively as well as strategically. In the pseudocode of Algorithm 2, the line "$y^t :=$ an optimal *best reply* to x^t" should be interpreted as follows: When Player I's strategy has been "frozen" to x^t, the resulting game is a one-player game for Player II, also known as an *absorbing Markov decision process*. For such a process, an optimal stationary strategy y^t that is pure is known to exist, and can be found in polynomial time using linear programming [15]. The expression maximin$(A_i(v^{t-1}))$ denotes a maximin mixed strategy (an "optimal strategy") for the maximizing row player in the matrix game $A_i(v^{t-1})$. This optimal strategy may again be found using linear programming. The strategy iteration algorithm was originally described for *one-player games* by Howard [15], with Player I being the single player – in that case, in the pseudocode, the line "$y^t :=$ an optimal *best reply* to x^t" is simply omitted. Subsequently, a variant of the pseudocode of Algorithm 2 was shown by Hoffman and Karp [14] to be a correct approximation algorithm for the class of *recurrent undiscounted stochastic games* and by Rao, Chandrasekaran and Nair [18] to be a correct algorithm for the class of *discounted stochastic games*. Finally, Chatterjee, de Alfaro and Henzinger [1] showed the pseudocode of Algorithm 2 to be a correct approximation algorithm for the class of reachability games. As is the case for value iteration, the strategy iteration algorithm is non-terminating, but has the property that as t approaches infinity, the valuations v_i^t approach the correct values v_i from below. Chatterjee *et al.* [1, Lemma 8] prove this by relating the algorithm to the value iteration algorithm. In particular, they prove:

$$\tilde{v}_i^t \leq v_i^t \leq v_i. \tag{1}$$

That is, strategy iteration needs at most as many iterations of the while loop as value iteration to achieve a particular degree of approximation to the correct values v_i. Also, the strategies x^t *guarantee* the valuations v_i^t for Player I, so whenever these valuations are ϵ-close to the values, the corresponding x^t is an ϵ-optimal strategy. However, until the present paper, there has been no published information on the number of iterations needed for the approximation to be an ϵ-optimal solution, though a recent breakthrough result of Friedman [9] proved that for the case of *turn-based* games, the number of iterations is at least exponential in N in the worst case. Clearly, the concurrent case is at least as bad. In fact, this paper will show that the concurrent case is much worse!

As our main result, we exhibit a family of reachability games with N positions and m actions for each player in each position, such that all non-terminal positions have value one and such that value iteration as well as strategy iteration need at least a *doubly exponential* $2^{m^{\Omega(N)}}$ number of iterations to obtain valuations larger than any fixed constant (say 0.01). By inequality (1), it is enough to consider the strategy iteration algorithm to establish this. However, our proof is much easier and cleaner for the value iteration algorithm, the exact bounds are somewhat better, and our much more technical proof for the strategy iteration case is in fact based upon it. So, we shall present separate proofs for the two cases, and in these proceedings, due to lack of space, only with details for the first case.

Our hard instances $P(N, m)$ for both algorithms are generalizations of the "Purgatory" games defined by Hansen, Miltersen and Koucký [12] (these occur as special cases by setting $m = 2$). Following the conventions of that paper, we describe these games as being games between *Dante* (Player I) and *Lucifer* (Player II). The game $P(N, m)$ can be described succinctly as follows: *Lucifer repeatedly selects and hides a number between 1 and m. Each time Lucifer hides such a number, Dante must try to guess which number it is. After the guess, the hidden number is revealed. If Dante ever guesses a number which is strictly higher than the one Lucifer is hiding, Dante loses the game. If Dante ever guesses correctly N times in a row, the game ends with Dante being the winner. If neither of these two events ever happen and the play thus continues forever, Dante loses.* It is easy to see that $P(N, m)$ can be described as a deterministic concurrent reachability game with N non-terminal positions and m actions for each player in each position. Also, by applying a polynomial-time algorithm by de Alfaro *et al.* [6] for determining which positions in a reachability game have value 1, we find that all positions except TRAP have value 1 in $P(N, m)$. That is, Dante can win this game with arbitrarily high probability.

We note that these hard instances are very natural and easy to describe "as games" that one might even conceivably have a bit of fun playing (the reader is invited to try playing $P(2, 2)$ with an uninitiated party)! In this respect, they are quite different from the recent extremely ingenious turn-based games due to Friedman [9] where strategy iteration exhibits exponential behavior.

Using recent improved upper bounds on the *patience* of ϵ-optimal strategies for Everett's recursive games, we provide matching $2^{m^{O(N)}}$ upper bounds on the number of iterations sufficient for getting adequate approximate values, by each of the algorithms. In particular, both algorithms are also of *at most* doubly-exponential complexity.

# Iterations	10^0	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8
Valuation	0.01347	0.03542	0.06879	0.10207	0.13396	0.16461	0.19415	0.22263	0.24828

Table 1. Running Strategy Iteration on $P(7, 2)$

That the doubly-exponential complexity is a real phenomenon is illustrated in Table 1 which tabulates the valuations computed by strategy iteration for the initial position of $P(7, 2)$, i.e., "Dante's Purgatory" [12], a 7-position game of value 1. The algorithm was implemented using floating point arithmetic and was allowed to run for one hundred million iterations at which point the precision was inadequate for representing the computed strategies (note that the main result of Hansen, Miltersen and Koucký [12] implies that roughly 64 *decimal* digits of precision is needed to describe a strategy achieving a valuation above 0.9).

Interestingly, when introduced as an algorithm for solving concurrent reachability games [1], strategy iteration was proposed as a practical alternative to generic algorithms having an exponential worst case complexity. More precisely, one obtains a generic algorithm for solving reachability games quantitatively by reducing the problem to the decision problem for the existential fragment of the first order theory of the real numbers [7]. This yields an exponential time (in fact a **PSPACE**) algorithm. Our results show that this generic algorithm is in fact astronomically more practical than strategy iteration on very simple and natural instances. Still, it is not practical in any real sense of this term, even given state-of-the-art implementations of the best known

decision procedures for the theory of the reals. Finding a practical algorithm remains a very interesting open problem.

1.2 Overview of proof techniques

Our proof of the lower bound for the case of value iteration is very intuitive. It is based on combining the following facts:

1. The valuations $\tilde{v}_i^t$ obtained in iteration t of value iteration is in fact the values of a *time bounded* version of the reachability game, where Player I loses if he has not reached GOAL at time t.
2. While the value of the game $P(N, m)$ is 1, the value of its time bounded version is very close to 0 for all small values of t.

The second fact was established by Hansen *et al.* [12] for the case $m = 2$ by relating the so-called *patience* of reachability games to the values of their time bounded version, without the connection to the value iteration algorithm being made explicit, by giving bounds on the patience of the games $P(N, 2)$. The present paper provides a different and arguably simpler proof of the lower bound on the value of the time bounded game that gives bounds also for other values of m than 2. It is based on exhibiting a fixed strategy for Lucifer that prevents Dante from winning fast.

The lower bound for strategy iteration is much more technical. We remark that the analysis of value iteration is used twice and in two different ways in the proof. It proceeds roughly as follows: The analysis of value iteration yields that when value iteration is applied to $P(1, m)$, exponentially many iterations (in m) are needed to yield a close approximation of the value. We can also show that when strategy iteration is applied to $P(1, m)$, exactly the same sequence of valuations is computed as when value iteration is applied to the same game. From these two facts, we can derive an upper bound on the patience of the strategies computed by strategy iteration on $P(1, m)$. Next, a quite involved argument shows that when applying strategy iteration to $P(N, m)$, the sequence of strategies computed for one of the positions (the initial one) is exactly the same as the one computed when strategy iteration is applied to $P(1, m)$. We also show that the smallest behavior probability in the computed strategy for $P(N, m)$ occurs in the initial position. In particular, the patiences of the sequence of strategies computed for $P(N, m)$ is the same as the patiences of the sequence of strategies computed for $P(1, m)$. Finally, our analysis of value iteration for $P(N, m)$ and the relationship between patience and value iteration allow us to conclude that a strategy with low patience for $P(N, m)$ cannot be near-optimal, yielding the desired doubly-exponential lower bound.

2 Theorems and Proofs

2.1 The connection between patience, the value of time bounded games, and the complexity of value iteration

The key to understanding value iteration is the following folklore lemma. Given a concurrent reachability game G, we define G_T to be the *finite* extensive form game with the

same rules as G, except that Player 1 loses if he has not reached GOAL after T moves of the pebble. The positions of G_T are denoted by (i, t), where i is a position of G and t is an integer denoting the number of time steps left until Dante's time is out.

Lemma 1. *The valuation $\tilde{v}_i^t$ computed by the value iteration algorithm when applied to a game G is the exact value of position (i, t) in the game G_t.*

The proof is an easy induction in t ("Backward induction"). A very general result by Mertens and Neyman [16] establishes that for a much more general class of games (undiscounted stochastic games), the value of the time bounded version converges to the value of the infinite version as the time bound approaches infinity. Combining this with Lemma 1 immediately yields the correctness of the value iteration algorithm.

The *patience* [8] of a stationary strategy for a concurrent reachability game is $1/p$, where p is the smallest non-zero behavior probability employed by the strategy in any position. The following lemma relates the patience of near-optimal strategies of a reachability game to the difference between the values of the time bounded and the infinite game and hence to the convergence rate of value iteration.

Lemma 2. *Let G be a reachability game with N non-terminal positions and with an ϵ-optimal strategy of patience at most l, for some $l \geq 1, \epsilon > 0$. Let $T = kNl^N$ for some $k \geq 1$, and u be any position of G. Then, the value of position (u, T) of G_T differs from the value of the position u of G by at most $\epsilon + e^{-k}$.*

Proof. We want to show that the value of (u, T) in G_T is at least $v_u - \epsilon - e^{-k}$, where v_u is the value of position u in G. We can assume that $v_u > \epsilon$, because otherwise we are done. Fix an ϵ-optimal stationary strategy x for Dante in G of patience at most l. Consider this as a strategy of G_T and consider play starting in u. We shall show that x guarantees Dante to win G_T with probability at least $v_u - \epsilon - e^{-k}$, thus proving the statement. Consider a best reply y by Lucifer to x in G_T. Note that y does not necessarily correspond to a stationary strategy in G. The strategy can still be played by Lucifer in G, by playing by it for the first T time steps and playing arbitrarily afterwards.

Call a position v of G *alive* if there are paths from v to GOAL in *all* directed graphs obtained from G in the following way: The nodes of the graphs are the positions of G. We then select for each position an arbitrary column for the corresponding matrix, and let the edges going out from this node correspond to the pointers of the chosen column and rows where Dante assigns positive probability. That is, intuitively, a position v is alive, if and only if there is no *absolutely sure* way for Lucifer for preventing Dante from reaching GOAL when play starts in v. Positions that are not alive are called *dead*. Note that if a position v is dead, the strategy y, being a best reply of Lucifer, will pick actions so that the probability of play reaching GOAL, conditioned on play having reached v, is 0. On the other hand, if the current position v is alive, the conditional probability that play reaches GOAL within the next N steps is at least $(1/l)^N$. That is, looking at the entire play, the probability that play has *not* reached either GOAL or a dead state after T steps is at most $(1 - l^{-N})^{T/N} = (1 - l^{-N})^{kl^N} \leq e^{-k}$. Suppose now that GOAL is reached in T steps with probability strictly less than $v_u - \epsilon - e^{-k}$ when play starts in u. This means that a dead position is reached with probability strictly greater than $1 - (v_u - \epsilon - e^{-k}) - e^{-k}$, i.e., strictly greater than $1 - (v_u - \epsilon)$. But this means that if

Lucifer plays y as a reply to x in the *infinite* game G he will in fact succeed in getting the pebble to reach a dead position and hence prevent Dante from *ever* reaching GOAL, with probability strictly greater than $1-(v_u-\epsilon)$. This contradicts x being ϵ-optimal for Dante in G. Thus, we conclude that GOAL is in fact reached in T steps with probability at least $v_u-\epsilon-e^{-k}$ when play starts in u with x and y being played against each other in G_T, as desired.

The connection between the convergence of value iteration and the time bounded version of the game allows us to reformulate the lemma in the following very useful way.

Lemma 3. *Let G be a reachability game with an ϵ-optimal strategy of patience at most l, for some $\epsilon > 0$. Then, $T = kNl^N$ rounds of value iteration is sufficient to approximate the values of all positions of the game with additive error at most $\epsilon + e^{-k}$.*

We can use this lemma to prove our upper bound on the number of iterations of value iteration (and hence also strategy iteration). The following lemma is from Hansen *et al.* [11].

Lemma 4 (Hansen, Koucký, Lauritzen, Miltersen and Tsigaridas). *Let $\epsilon > 0$ be arbitrary. Any concurrent reachability game with N positions and at most $m \geq 2$ actions in each position has an ϵ-optimal stationary strategy of patience at most $(1/\epsilon)^{m^{O(N)}}$.*

This lemma is an asymptotic improvement of Theorem 4 of Hansen *et al.* [12], that gave an upper bound of $(1/\epsilon)^{2^{30M}}$, for a *total* number of M actions, when $M \geq 10$ and $0 < \epsilon < \frac{1}{2}$. This result does however have the advantage of an *explicit* constant in the exponent, which the bound of Lemma 4 lacks.

Combining Lemma 3, Lemma 4, and also applying inequality (1), we get the following upper bound:

Theorem 5. *Let $\epsilon > 0$ be arbitrary. When applying value iteration or strategy iteration to a concurrent reachability game with N non-terminal positions and $m \geq 2$ choices for each player in each position, after at most $(1/\epsilon)^{m^{O(N)}}$ iterations, an ϵ-approximation to the value has been obtained.*

Also, Lemma 3 will be very useful for us below when applied in the contrapositive. Specifically, below, we will directly analyze and compare the value of $P(N,m)$ with the value of its time bounded version, and use this to conclude that the value iteration algorithm does not converge quickly when applied to this game. The lemma then implies that the patience of any ϵ-optimal strategy is large. When we later consider the strategy iteration algorithm applied to the same game, we will show that the strategy computed after any sub-astronomical number of iterations has too low patience to be ϵ-optimal.

2.2 The value of time bounded Generalized Purgatory and the complexity of value iteration

In this section we give an upper bound on the value of a time bounded version of the Generalized Purgatory game $P(N,m)$. As explained in Section 2.1, this upper bound

immediately implies a lower bound on the number of iterations needed by value iteration to approximate the value of the original game.

We let $P_T(N,m)$ be the time bounded version of $P(N,m)$ as defined in Section 2.1, i.e. $P_T(N,m)$ is syntactic sugar for $(P(N,m))_T$. Also, we need to fix an indexing of the positions of $P(N,m)$. We define position i for $i = 1,\ldots,N$ to be the position where Dante already guessed correctly $i-1$ times in a row and still needs to guess correctly $N-i+1$ times in a row to win the game.

First we give a rather precise analysis of the one-position case. Besides being interesting in its own right (to establish that value iteration is exponential even for this case), this will also be useful later when we analyze strategy iteration.

Theorem 6. *Let $m \geq 2$ and $T \geq 1$. The value of position $(1,T)$ of $P_T(1,m)$ is less than*

$$1 - (1 - \frac{1}{m})(\frac{1}{mT})^{1/(m-1)}.$$

Proof. Let $\epsilon = (1/mT)^{1/(m-1)}$. Consider any strategy (not necessarily stationary) for Dante for playing $P_T(1,m)$. In each round of play, Dante chooses his action with a probability distribution that may depend on previous play and time left. We define a reply by Lucifer in a round-to-round fashion.

Fix a history of play leading to some current round and let $p_1, p_2, \ldots, p_m$ be the probabilities by which Dante plays $1, 2, \ldots, m$ in this current round. There are two cases.

1. There is an i so that $p_i < (\frac{1-\epsilon}{\epsilon}) \sum_{j \geq i+1} p_j$. We call such a round a *green* round. In this case, Lucifer plays i.
2. For all i, $p_i \geq (\frac{1-\epsilon}{\epsilon}) \sum_{j \geq i+1} p_j$. We call such a round a *red* round. In this case, Lucifer plays m.

This completes the definition of Lucifer's reply.

We now analyze the probability that Dante wins $P_T(1,m)$ when he plays his strategy and Lucifer plays this reply. We show this probability to be at most

$$1 - (1 - \frac{1}{m})(\frac{1}{mT})^{1/(m-1)}$$

and we shall be done.

Let us consider a green round. We claim that the probability that Dante wins in this round, conditioned on the previous history of play, and conditioned on play ending in this round, is at most $1-\epsilon$. Indeed, this conditional probability is given by

$$\begin{aligned} \frac{p_i}{p_i + (p_{i+1} + \cdots + p_m)} &< \frac{(\frac{1-\epsilon}{\epsilon})(\sum_{j \geq i+1} p_j)}{(\frac{1-\epsilon}{\epsilon})(\sum_{j \geq i+1} p_j) + (\sum_{j \geq i+1} p_j)} \\ &= \frac{(1-\epsilon)/\epsilon}{(1-\epsilon)/\epsilon + \epsilon/\epsilon} \\ &= 1 - \epsilon. \end{aligned}$$

Let us next consider a red round. We claim that the probability of play ending in this round, conditioned on the previous history of play, is at most ϵ^{m-1}. Indeed, note that this conditional probability is exactly p_m, and that

$$1 = \sum_{j=1}^{m} p_j = p_1 + \sum_{j=2}^{m} p_j \geq (1 + \frac{1-\epsilon}{\epsilon})(\sum_{j=2}^{m} p_j) = (1 + \frac{1-\epsilon}{\epsilon})(p_2 + \sum_{j=3}^{m} p_j)$$

$$\geq (1 + \frac{1-\epsilon}{\epsilon})^2(\sum_{j=3}^{m} p_j) \geq \cdots \geq (1 + \frac{1-\epsilon}{\epsilon})^{m-1} p_m = (\frac{1}{\epsilon})^{m-1} p_m$$

from which $p_m \leq \epsilon^{m-1}$. That is, in every round of play, conditioned on previous play, either it is the case that the probability that play ends in this round is at most ϵ^{m-1} (for the case of a red round) or it is the case that conditioned on play ending, the probability of win for Dante is at most $1 - \epsilon$ (for the case of a green round).

Now let us estimate the probability of a win for Dante in the entire game $P_T(1, m)$. Let W denote the event that Dante wins. Let G be the event that play ends in a green round. Also, let R be the event that play ends in a red round. Then, we have

$$\begin{aligned}
\Pr[W] &= \Pr[W|R]\Pr[R] + \Pr[W|G]\Pr[G] \\
&\leq \Pr[R] + \Pr[W|G]\Pr[G] \\
&= \Pr[R] + \Pr[W|G](1 - \Pr[R]) \\
&= \Pr[R] + \Pr[W|G] - \Pr[R]\Pr[W|G] \\
&< (\epsilon^{m-1})T + (1-\epsilon) - (\epsilon^{m-1})T(1-\epsilon) \\
&= 1 - \epsilon + T\epsilon^m \\
&= 1 - (\frac{1}{mT})^{1/(m-1)} + T(\frac{1}{mT})^{\frac{m}{m-1}} \\
&= 1 - (1 - \frac{1}{m})(\frac{1}{mT})^{1/(m-1)}.
\end{aligned}$$

Combining Lemma 1 with Theorem 6 we get the result that value iteration needs exponential time, even for one-position games.

Corollary 7. *Let $0 < \epsilon < 1$. Applying less than $\frac{1}{em}(1/\epsilon)^{m-1}$ iterations of the value iteration algorithm to $P(1, m)$ yields a valuation at least ϵ smaller than the exact value.*

Next, we analyze the N-position case, where we give a somewhat coarser bound.

Theorem 8. *Let N, m, k, T be integers with $N \geq 2, m \geq 2, 1 \leq k \leq N-2$ and $T \leq 2^{m^{N-k}}$. Then, the value of $P_T(N, m)$ is at most $2m^{-k} + 2^{-m^{N-k-1}}$.*

Proof. We show an upper bound on the value of $P_T(N, m)$ of $2m^{-k} + 2^{-m^{N-k-1}}$ by exhibiting a particular strategy of Lucifer and showing that any response by Dante to this particular strategy of Lucifer will make Dante win with probability at most $2m^{-k} + 2^{-m^{N-k-1}}$.

To structure the proof, we divide the play into *epochs*. An epoch begins and another ends immediately after each time Dante has guessed incorrectly by undershooting, so

that he now finds himself in exactly the same situation as when the play begins (but in general with less time left to win). That is, Dante wins if and only if there is an epoch of length N containing only correct guesses. For convenience, we make the game a little more attractive for Dante by continuing play for T epochs, rather than T rounds. Call this prolonged game G'_T. Clearly, the value of G_T is at most the value of G'_T, so it is okay to prove the upper bound for the latter. We index the epochs $1, 2, \ldots, T$.

To define the strategy of Lucifer, we first define a function $f : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ as follows:

$$f(i, j) = 1 + (j - 1) \sum_{r=0}^{i-1} m^r.$$

Then, it is easy to see that f satisfies the following two equations.

$$f(i, m) = m^i \tag{2}$$

$$f(i, j + 1) = f(i, j) + \sum_{r=0}^{i-1} f(r, m) \tag{3}$$

The specific strategy of Lucifer is this: Let d be the number of rounds already played in the current epoch. If $d \geq N - k$, Lucifer chooses a number between 1 and m uniformly at random. If $d < N - k$, he hides the numbers $j = 1, \ldots, m - 1$ with probabilities $p_j(d) = 2^{-f(N-k-d,m+1-j)}$ and puts all remaining probability mass on the number m (since $N - k - d \geq 1$ and $m \geq 2$, there is indeed some probability mass left for m).

Freeze the strategy of Lucifer to this strategy. From the point of view of Dante, the game G_T is now a finite horizon absorbing Markov decision process. Thus, he has an optimal policy that is deterministic and history independent. That is, the choices of Dante according to this policy depend only on the number of rounds already played in the present epoch and the remaining number of epochs before the limit of T epochs has been played, or, equivalently, on the index of the current epoch. We can assume without loss of generality that Dante plays such an optimal policy. That is, his optimal policy for epoch t can be described by a specific sequence of actions $a_{t0}, a_{t1}, a_{t2}, \ldots, a_{t(N-1)}$ in $\{1, \ldots, m\}$ to make in the next N rounds (with the caveat that this sequence of choices will be aborted if the epoch ends).

Se define the following mutually exclusive events W_t, L_t:

- W_t: Dante wins the game in epoch t (by guessing correctly N times).
- L_t: Dante loses the game in epoch t (by overshooting Lucifer's number)

We make the following claim:

Claim: For each t, either $\Pr[W_t] \leq 2^{-m^{N-k}-m^{N-k-1}}$ or $\Pr[W_t]/\Pr[L_t] \leq 2m^{-k}$.

First, let us see that the claim implies the lemma. Indeed, the probability of Dante winning can be split into the contributions from those epochs where Dante wins with probability at most $2^{-m^{N-k}-m^{N-k-1}}$ and the remaining epochs. The total winning probability mass from the first is at most $T2^{-m^{N-k}-m^{N-k-1}} \leq 2^{-m^{N-k-1}}$ and the total winning probability mass of the rest is at most $2m^{-k}$, giving an upper bound for Dante's winning probability of $2m^{-k} + 2^{-m^{N-k-1}}$.

So let us prove the claim. Fix an epoch t and let $a_{t0}, a_{t1}, a_{t3}, \ldots, a_{t(N-1)}$ be Dante's sequence of actions. Suppose $a_{t0} = 1$ and $a_{t1} = 1$. Then, since Lucifer only plays 1 in the first two rounds with probability $p_1(0)p_1(1) = 2^{-f(N-k,m)} \cdot 2^{-f(N-k-1,m)}$, Dante only wins the game in this epoch with at most that probability, which by equation (2) is equal to $2^{-m^{N-k}-m^{N-k-1}}$, as desired.

Now assume $a_{t0} > 1$ or $a_{t1} > 1$. We want to show that $\Pr[W_t]/\Pr[L_t] \leq 2m^{-k}$. Let d be the largest index so that $d < N - k$ and so that $a_{td} > 1$. Since $a_{t0} > 1$ or $a_{t1} > 1$, such a d exists. Let E be the event that epoch t lasts for at least d rounds. We will show that $\Pr[W_t|E]/\Pr[L_t|E] \leq 2m^{-k}$. Since $W_t \subseteq E$, this also implies that $\Pr[W_t]/\Pr[L_t] \leq 2m^{-k}$. Since we condition on E we look at Dante's decision after d rounds of epoch t. He chooses the action $j = a_{td} > 1$. If Lucifer at this point chooses a number small than j, Dante loses. In particular, since Lucifer chooses the number $j-1$ with probability $2^{-f(N-k-d,m+1-(j-1)}$, Dante loses the entire game by his action a_{td} with probability at least $2^{-f(N-k-d,m-j)}$, conditioned on E. On the other hand the probability that he wins the game in this epoch conditioned on E is at most $(2^{-f(N-k-d,m+1-j)})(\prod_{i=d+1}^{N-k-1} 2^{-f(N-k-i,m)})(m^{-k}))$, the first factor being the probability that Lucifer chooses j at round d, the second factor being the probability that Lucifer like Dante repeatedly chooses 1 until the last k rounds of the epoch begin, and the third factor being the probability that Lucifer matches Dante's choices in those k rounds. Now we have

$$\begin{aligned}
\Pr[W_t]/\Pr[L_t] &\leq \\
\Pr[W_t|E]/\Pr[L_t|E] &\leq \\
(2^{-f(N-k-d,m+1-j)})(\prod_{i=d+1}^{N-k-1} 2^{-f(N-k-i,m)})(m^{-k}))2^{f(N-k-d,m-j)} &\leq \\
m^{-k}2^{f(N-k-d,m-j)-f(N-k-d,m+1-j)-\sum_{r=1}^{N-k-d-1} f(r,m)} &= \\
2m^{-k}2^{f(N-k-d,m-j)-f(N-k-d,m+1-j)-\sum_{r=0}^{N-k-d-1} f(r,m)} &= \\
2m^{-k} &
\end{aligned}$$

as desired.

Combining Lemma 1 with Theorem 8 we get the result that value iteration needs doubly exponential time to obtain any non-trivial approximation:

Corollary 9. *Let N be even. Applying less than $2^{m^{N/2}}$ iterations of the value iteration algorithm to $P(N,m)$ yields a valuation of the initial position of at most $3m^{-N/2}$, even though the actual value of the game is 1.*

We also get the following bound on the patience of near-optimal strategies of $P(N,m)$ that will be useful when analyzing strategy iteration.

Theorem 10. *Suppose N is sufficiently large and $m \geq 2$. Let $\epsilon = 1 - 4m^{-N/2}$. Then all ϵ-optimal strategies of $P(N,m)$ have patience at least $2^{m^{N/3}}$.*

Proof. Putting $c = \frac{N \ln m}{2}$, Lemma 2 tells us that if $P(N,m)$ has an ϵ-optimal strategy of patience less than $l = 2^{m^{N/3}}$, then the value of $P_t(N,m)$ is at least $1 - \epsilon - e^{-c} =$

$3m^{-N/2}$, where $t = cNl^N \leq 2^{m^{N/2}}$. But putting $k = N/2$, Theorem 8 tells us that the value of $P_t(N, m)$ is at most $2m^{-N/2} + 2^{-m^{N/2-1}} < 3m^{-N/2}$, a contradiction.

2.3 Strategy Iteration

The technical content of this section is a number of lemmas on what happens when the strategy iteration algorithm is applied to $P(N, m)$, leading up to the following crucial lemma:

Lemma 11. *When applying strategy iteration to $P(N, m)$, the patience of the strategy x^t computed in iteration t is at most $e \cdot m \cdot t$.*

Before we prove Lemma 11, we show that it implies the lower bound we are looking for.

Theorem 12. *Suppose N is sufficiently large. Applying less than $2^{m^{N/4}}$ iterations of strategy iteration to $P(N, m)$ yields a valuation of the initial position of less than $4m^{-N/2}$, despite the fact that the value of the position is 1.*

Proof. Lemma 11 implies that the patience of the strategy x^t computed in iteration t for $t = 2^{m^{N/4}}$ is at most $em2^{m^{N/4}}$. Theorem 10 states that if $\epsilon = 1 - 4m^{-N/2}$, then all ϵ-optimal strategies of $P(N, m)$ have patience at least $2^{m^{N/3}}$. So x^t is not ϵ-optimal and the bound follows.

Due to space constraints, the rather long and technical proof of Lemma 11 itself is omitted, but can be found in the full version of this paper [10].

Acknowledgement

First and foremost, we would like to thank Uri Zwick for extremely helpful discussions and Kousha Etessami for being instrumental for starting this research. We would also like to thank Vladimir V. Podolskii for helpful discussions.

References

1. Chatterjee, K. de Alfaro, L., Henzinger, T. A: Strategy improvement for concurrent reachability games. In: Third International Conference on the Quantitative Evaluation of Systems., pp. 291–300. IEEE Press, New York (2006)
2. Chatterjee, K., de Alfaro, L, Henzinger, T. A.: Termination criteria for solving concurrent safety and reachability games. In: 20th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 197–206. SIAM, Philadelphia (2009)
3. Chatterjee, K. Majumdar, R. Jurdziński, M.: On Nash equilibria in stochastic games. In: Marcinkowski, J. and Tarlecki, A. (eds.) CSL 2004. LNCS, vol. 3210, pp. 26–40. Springer, Heidelberg (2004)
4. Condon, A.: On algorithms for simple stochastic games. Advances in Computational Complexity Theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 13, 51–73 (1993)

5. Dai, D. and Ge, R. New results on simple stochastic games. In: Dong, Y., Du, D.-Z., Ibarra,O.H. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 1014–1023. Springer, Heidelberg (2009)
6. de Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. Theor. Comput. Sci. 386,188–217 (2007)
7. Etessami, K. and Yannakakis, M.: Recursive concurrent stochastic games. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I.: (eds.) ICALP 2006 (2), LNCS, vol. 4052, pp. 324–335. Springer, Heidelberg (2006)
8. Everett, H.: Recursive games. In: Kuhn, H.W., Tucker, A.W. (eds.) Contributions to the Theory of Games Vol. III, volume 39 of Annals of Mathematical Studies. pp. 47–78, Princeton University Press, Princeton (1957)
9. Friedmann, O.: An exponential lower bound for the parity game strategy improvement algorithm as we know it. In: 24th Annual IEEE Symposium on Logic in Computer Science, pp. 145–156. IEEE Press, New York (2009)
10. Hansen, K.A., Ibsen-Jensen, R., Miltersen, P.B.: The complexity of solving reachability games using value and strategy iteration, `http://arxiv.org/abs/1007.1812`
11. Hansen, K.A., Koucký, M., Lauritzen, N., Miltersen, P.B., Tsigaridas, E.: Exact algorithms for solving discounted stochastic games and recursive games. In 43rd ACM Symposium on Theory of Computing, to appear, ACM Press, New York (2011)
12. Hansen, K.A., Koucký, M., Miltersen, P.B.: Winning concurrent reachability games requires doubly exponential patience. In: 24th Annual IEEE Symposium on Logic in Computer Science, pp. 332–341. IEEE Press, New York (2009)
13. Himmelberg, C.J., Parthasarathy, T., Raghavan, T.E.S., Vleck, F.S.V.: Existence of p-equilibrium and optimal stationary strategies in stochastic games. Proc. Amer. Math. Soc. 60, 245–251 (1976)
14. Hoffman, A., Karp, R.: On nonterminating stochastic games. Management Science 12, 359–370 (1966)
15. Howard, R.: Dynamic Programming and Markov Processes. MIT Press, Cambridge, Massachusetts (1960)
16. Mertens, J.F., Neyman, A.: Stochastic games. International Journal of Game Theory, 10, 53–66 (1981)
17. Parthasarathy, T.: Discounted and positive stochastic games. Bull. Amer. Math. Soc. 77, 134–136 (1971)
18. Rao, S., Chandrasekaran, R., Nair, K.. Algorithms for discounted games. J. Optimiz. Theory App. 11, 627–637 (1973)
19. Shapley, L.S.. Stochastic games. Proceedings of the National Academy of Sciences, U.S.A. 39, 1095–1100 (1953)