On Finite Strategy Sets for Finitely Repeated Zero-Sum Games[*]

Thomas C. O'Connell[†‡]
Department of Mathematics and Computer Science
Skidmore College
815 North Broadway
Saratoga Springs, NY 12866
E-mail: `oconnellT@acm.org`

Richard E. Stearns
Department of Computer Science
University at Albany, State University of New York
Albany, NY 12222
E-mail: `res@cs.albany.edu`

January, 2003

Abstract

We study finitely repeated two-person zero-sum games in which Player 1 is restricted to mixing over a fixed number of pure strategies while Player 2 is unrestricted. We describe an optimal set of pure strategies for Player 1 along with an optimal mixed strategy. We show that the entropy of this mixed strategy appears as a factor in an exact formula for the value of the game and thus is seen to have a direct numerical effect on the game's value. We develop upper and lower bounds on the value of these games that are within an additive constant and discuss how our results are related to the work of Neyman and Okada on strategic entropy (Neyman and Okada, 1999, *Games Econ. Behavior* **29**, 191-223). Finally, we use these results to bound the value of repeated games in which one of the players uses a computer with a bounded memory and is further restricted to using a constant amount of time at each stage. *Journal of Economic Literature* Classification Number: C72

Key Words: Bounded rationality, entropy, repeated games, finite automata

1   Introduction

There have been many studies of bounded rationality in repeated games (see Kalai (1993) and Aumann (1997) for surveys). A number of these studies restrict one or more of the players to pure strategies that can be implemented on a finite automaton of a particular size where the size is the number of states in the automaton. (See Neyman (1997) for a thorough discussion.) Neyman and Okada (1999), for instance, consider N-stage two-person zero-sum games in which Player 1 is restricted to pure strategies which can be implemented on a finite automaton of size $m(N)$ while Player 2 is unrestricted. They show that if $m(N)\log_2 m(N) = o(N)$ then Player 1's expected payoff converges to his stage game maxmin value as N goes to infinity. (We assume Player 1 is the maximizer throughout this paper.)

The use of the finite automata model is very appealing because finite automata generate actions in a timely manner (namely in one step), they have limited computing power, and they do not require additional resources once they are started. However, the number of states in an automaton is (except for its simplicity) a very poor model of the automaton's complexity. This is because the amount of hardware [1] needed to implement an n-state automaton can vary exponentially with the automaton's description. To see this, consider the following example of three strategies for playing matching pennies:

Example 1.1 *Let n be the stage number, a_n be the action chosen by Player 1 at stage n and b_n be the action chosen by Player 2 at stage n.*

Strategy A: *Play $a_n = H$ if $n \equiv 0 \bmod 2^{20}$ and $a_n = T$ otherwise.*

Strategy B: *Let R be a random table of 2^{20} H's and T's. In other words, each element of the table is an H with probability $1/2$ and a T with probability $1/2$. Play $a_n = R[n \bmod 2^{20}]$ where $R[i]*$

---

[1]The "amount of hardware" could be formally defined as the number of gates and registers needed to implement the automaton as a sequential circuit, but we do not need such formality here.

is the i-th element of the table.

Strategy C: *Play $a_n = H$ if $n \leq 2^{20}$ and $a_n = b_{n-2^{20}}$ otherwise.*

Consider the hardware required to implement these three strategies using computer programs. Given the simplicity of the programs and the computer's CPU, the "amount of memory" required by the programs accurately reflects the hardware requirements for these three examples where the amount of memory required by a program is the maximum number of bits of information that a computer must maintain to execute the program. (See Hopcroft and Ullman (1979) for details.) The obvious computer program to implement Strategy A needs only twenty bits of memory (not counting memory needed to store the program). Strategy B requires over a million bits of memory since it must remember the 2^{20} random choices. Yet, considered as finite automata, Strategies A and B each have 2^{20} states. This illustrates the "exponential variation" discussed above.

Strategy C requires the same amount of memory as Strategy B, this time to remember the last 2^{20} moves of the opponent. The implementation of Strategies B and C are virtually identical except that B uses the memory to store a fixed sequence whereas C changes the memory as the game unfolds. In other words, Strategy B could use a ROM (read only memory) whereas Strategy C requires RAM (random access memory). Although intuitively very similar, the two strategies have radically different complexity from a state counting point of view. Strategy B uses 2^{20} states whereas strategy C uses $2^{2^{20}}$ strategies, an exponential difference.

Theoretical results on relative complexity can be model dependent. For example, we say Player 2 *defeats* Player 1 if the value of the repeated game approaches the maxmin value of the stage game as the number of stages increases. Ben-Porath (1993) showed that if Player 1 is allowed to mix over all n-state finite automata then Player 2 needs exponentially more states to defeat Player 1. However, if we look at the proof of this result in terms of memory requirements, the proof has Player 1 mixing over finite automata which are simple cycles. These automata require n bits of memory

to implement. Since a finite automaton of size 2^n might require as little as n bits of memory to implement, this proof does not imply Player 2 needs more than n bits of memory to defeat Player 1. The proof, therefore, does not show that Player 2 needs exponentially more hardware or memory.

In contrast, Stearns (1989) shows that an $O(n)$-bit memory is all that Player 2 needs to defeat Player 1 if Player 1 is restricted to an n-bit memory, a linear increase. Because Stearns (1989) implements moves of a finite automaton with long computation sequences, the strategies are extremely impractical, but the point about complexity dependence is made.

Bounded rationality suggests that the players be limited to finite strategy sets. Bounding complexity is an appealing method of obtaining finite sets, but, as indicated by the above discussion, it is not clear what kind of "complexity" should be bounded. It is also possible that finite sets might be obtained using computer models other than finite automata. In this paper, we want to side step the issue of where finite strategy sets come from and study only the implications of finiteness. To this end, we consider repeated games in which Player 1 is allowed to mix over any K pure strategies for some fixed K.

In Section 2, we first determine Player 1's best set of K strategies given an unbounded opponent. One such set has a simple description which yields a computationally simple implementation of the strategies in the set. In particular, we show that, for any $K \geq 1$, Player 1 has an optimal set of K pure strategies which ignore Player 2's actions and, therefore, can be represented by a tree with K leaves. In Section 3, we construct an optimal mixed strategy for Player 1 when the stage game is 2×2. A behavorial view of this strategy has Player 1 playing his stage game optimal mixed strategy at every stage in which uncertainty remains as to which pure strategy Player 1 has chosen for the repeated game. In other words, Player 1 plays a locally optimal mixed strategy. We use Player 1's optimal mixed strategy to determine the value of the game. A significant feature of our equation for the value of the game is that the entropy of Player 1's optimal mixed strategy appears directly

in the equation. Thus this equation singles out entropy as the natural way to measure information. In Section 4, we consider stage games that are larger than 2×2. We show that in this case Player 1 may play locally suboptimal strategies in order to create additional uncertainty about his future play. Our analysis gives us a measure of the tradeoff Player 1 faces between ensuring a good payoff at the current stage and creating uncertainty about his future play. Again, entropy is used directly in this analysis. In this case, the entropy of several alternative strategies is considered. Using the results of Shapley and Snow (1950), we show how Player 1 can construct an approximately optimal set. We provide upper and lower bounds on the total expected payoff achievable by Player 1 in the repeated game. These bounds are within an additive constant independent of the length of the game and the number of pure strategies available to Player 1. Finally, in Section 6, we consider how Player 1 might go about implementing his strategies and how much computational resources he would require. We explain how a computer, given a description of a strategy from an optimal set, can execute the strategy using only a constant amount of time at each stage and a constant amount of memory beyond that required to hold the strategy's description. The length of this description is within a constant factor of the optimal description length. We also provide upper and lower bounds on the value of the repeated game in which Player 1 is restricted to a computer with a bounded memory and is further resricted to using a constant amount of time at each stage.

Related Work

Our interest in this area was motivated by Neyman and Okada's work on strategic entropy. In Neyman and Okada (1999), they introduced strategic entropy and used it as a technical tool to prove results on repeated games with finite automata and bounded recall.[2] Neyman and Okada (2000a) briefly mentions that strategic entropy could also be viewed as a "measure of the cost of randomization" and, therefore, could be thought of as a true measure of strategic complexity.

---

[2]It should be noted that **?**) first introduced the concept of entropy in the context of repeated games with bounded recall. This reference was regrettably not present in the final journal version of this paper.

Although Neyman and Okada make it clear that strategic entropy is a useful concept, there is no compelling reason to believe that some other measure might not be equally useful and more natural. Neyman and Okada (2000b) explicitly considers games with finite strategy sets but does not provide the level of detail presented here. In our analysis, entropy appears as a factor in an exact formula for the value of the game and thus is shown to have a direct numerical effect on the value.

Notation

We use the following notation in the remainder of this paper. A_i denotes the set of pure actions in the stage game available to Player i. S and T denote the set of pure strategies in the repeated game available to Players 1 and 2 respectively. $\Delta(S)$ is the set of mixed strategies over pure strategy set S. $|S|$ is the number of elements in set S. When Players 1 and 2 choose actions a_1 and a_2 respectively, the payoff in the stage game is denoted $r(a_1, a_2)$. Player 1's stage game maxmin value in pure actions is denoted by r_*, i.e. $r_* = \max_{a_1 \in A_1} \min_{a_2 \in A_2} r(a_1, a_2)$. $V(G)$ is the value of the stage game. $V(G^N)$ is the value of the N-stage repetition of G, i.e. $V(G^N) = \max_{\alpha \in \Delta(S)} \min_{t \in T} E_{\alpha,t} \left[\frac{1}{N} \sum_{n=1}^{N} r\left(a_1^n, a_2^n\right) \right]$ where a_i^n is the action chosen by Player i at stage n and $E_{\alpha,t}$ is the expected value given strategies α and t. We will also sometimes refer to the **optimal total expected payoff** for an N-stage game which is defined to be $\max_{\alpha \in \Delta(S)} \min_{t \in T} E_{\alpha,t} \left[\sum_{n=1}^{N} r\left(a_1^n, a_2^n\right) \right]$, namely the N-stage value before dividing by N. G_K^N represents the N-stage repetition of G when Player 1 is restricted to mixing over K pure strategies.

2   Representing an Optimal Set

Consider a finitely repeated two-person zero-sum game, G_K^N, in which Player 1 is restricted to picking a set of K pure strategies and then mixing over these strategies. Player 2 is informed of the set prior to the start of the game so there is no point in Player 1 mixing over the choice of sets.

Player 2 is not restricted in any way. In this section, we show that Player 1 can play optimally by choosing a set of pure strategies that ignore Player 2's actions. To see why this is true, consider the situation that occurs after the player's make their first actions a_1 and b_1 in G_K^N. This situation is identical to the game $G_{K_{a_1}}^{N-1}$ where K_{a_1} is the number of pure strategies available to Player 1 that begin with a_1. By "identical", we mean having the same available actions and having the same payoffs. Since the subgame can be described without reference to the action b_1 of Player 2, there is no advantage to either player knowing that b_1 was previously played. We prove this result formally in Theorem 2.1.

We call a pure strategy that ignores the opponent's actions **oblivious**. Player 1's optimal set of K oblivious strategies can be represented by a tree with K leaves. The simple structure of this tree enables Player 1 to compute the set along with the corresponding optimal mixed strategy using $O(K)$ time when the stage game is 2×2.

Definition 2.1 *A set of pure strategies S is an **optimal set** for Player 1 in G_K^N if $|S| \leq K$ and there is a mixed strategy $\sigma \in \Delta(S)$ such that, for all sets of pure strategies X with $|X| \leq K$,*

$$\min_{\tau \in \Delta(T)} E_{\sigma,\tau} \left[\frac{1}{N} \sum_{n=1}^{N} r\left(a_1^n, a_2^n\right) \right] \geq \max_{\gamma \in \Delta(X)} \min_{\tau \in \Delta(T)} E_{\gamma,\tau} \left[\frac{1}{N} \sum_{n=1}^{N} r\left(a_1^n, a_2^n\right) \right]$$

Suppose, for an N-stage game, Player 1 chooses a set of K oblivious pure strategies. Each strategy in the set can be described by a string of length N specifying the action to be taken at each stage. We can describe the entire set of strategies using a tree of depth N with K leaves where each leaf represents one of the strategies from the set. The labels on the edges of the path from the root to a leaf indicate the action chosen at that stage by the corresponding strategy. For example, suppose $A_1 = \{L, R\}$. The tree in Figure 1 describes the set $\{LLLL, LLLR, LLRL, LRLL, LRRL, RLLL, RLRL, RRLL\}$.

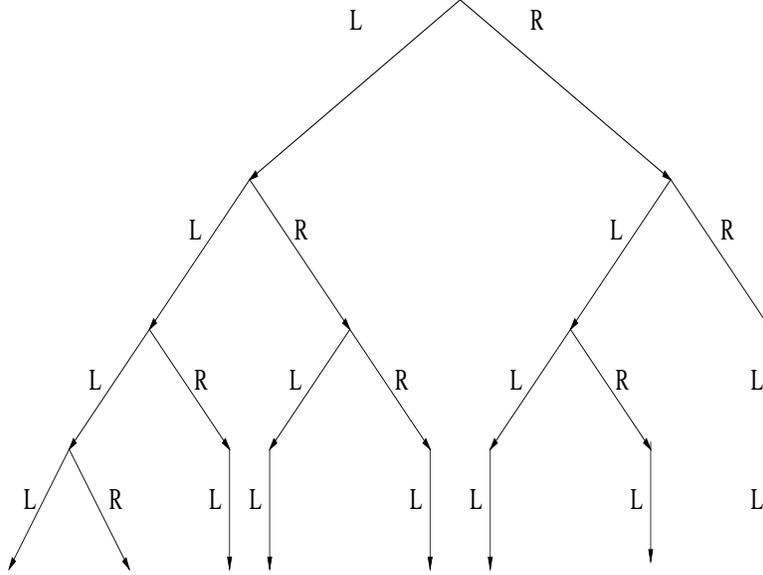We call a node of out-degree greater than one a **fork**. Suppose that, in the tree representation

Figure 1: Tree representation of {LLLL,LLLR,LLRL,LRLL,LRRL,RLLL,RLRL,RRLL}.

of Player 1's set of pure strategies, there is no path containing a node of out-degree one followed by a fork. Suppose further that every edge from the last fork to the leaf in any path is labeled with one of Player 1's stage game maxmin actions. In representing this set, we don't need to specify the actions after the last fork in a path since it is understood that a maxmin action is played from that stage onward. Under these assumptions, every internal node in the tree will be a fork. We say that a set of pure strategies has a **simple tree representation** if it can be represented by a tree in which every internal node is a fork and it is understood that once a leaf is reached the strategy plays one of Player 1's stage game maxmin actions from that stage onward.

We claim that Player 1 has an optimal set of pure strategies for G_K^N with a simple tree representation.

Theorem 2.1 *For any N and K, Player 1 has an optimal set of pure strategies for G_K^N that has a simple tree representation.*

Proof. First, we argue that Player 1 can play optimally while ignoring Player 2's actions. Consider a tree representation of a non-oblivious set of pure strategies. In this tree, we represent

7

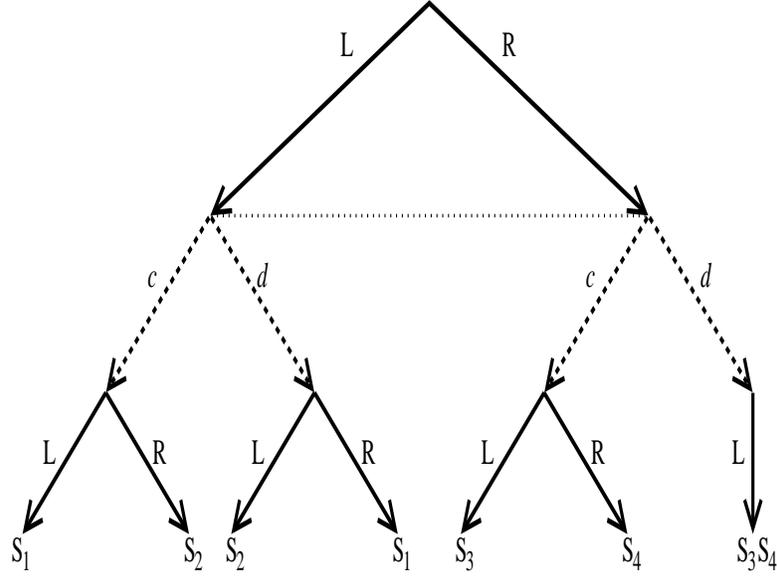Player 2's actions by forks with branches that are dashed lines (see Figure 2). For simplicity, assume

Figure 2: A tree representation of a non-oblivious set of strategies $\{S_1 = [L, ((c, L), (d, R))], S_2 = [L, ((c, R), (d, L))], S_3 = [R, ((c, L), (d, L))], S_4 = [R, ((c, R), (d, L))]\}$. The dotted horizontal line indicates Player 2's information set.

that each player has two actions. The following argument can easily be generalized.

Suppose there is some node in the tree underneath which is a subtree T_i^c if the action pair chosen at the previous stage was (a_i, c) and T_i^d if the action pair chosen at the previous stage was (a_i, d) for $i = 1, 2$ (see Figure 3). Let h be the history of play that lead to this point in the tree. For each i, let r_i^c and r_i^d be the expected payoffs in T_i^c and T_i^d respectively given that Player 1 plays an optimal mixed strategy ρ and Player 2 plays a best response to ρ. Since Player 2 is unrestricted, he can play a best response to ρ in any subgame. Therefore, after history h, Player 1 receives an expected payoff of

$$\min\{ \quad \rho(a_1|h)(r_1^c + r(a_1, c)) \quad + \quad \rho(a_2|h)(r_2^c + r(a_2, c)),$$

$$\rho(a_1|h)(r_1^d + r(a_1, d)) \quad + \quad \rho(a_2|h)(r_2^d + r(a_2, d)) \quad \} \tag{1}$$

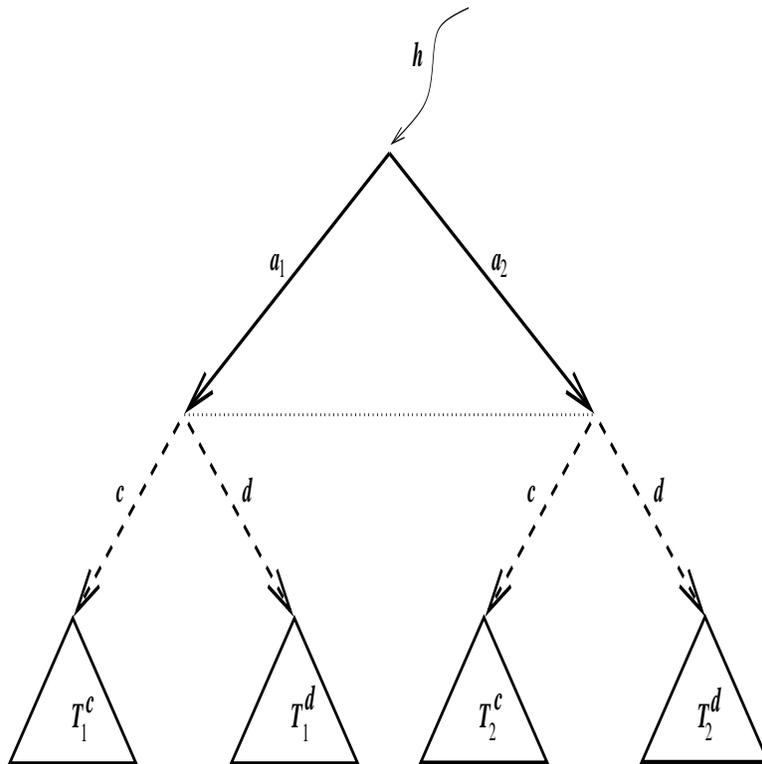Assume without loss of generality that $r_1^c \geq r_1^d$. If, after playing a_1 at the previous stage, Player 1

Figure 3: A tree showing Player 1 basing his strategy on Player 2's action after some history of play h. The dotted horizontal line indicates Player 2's information set.

plays T_1^c regardless of whether Player 2 plays c or d, his payoff is

$$\min\{\ \ \rho(a_1|h)(r_1^c + r(a_1, c))\ \ +\ \ \rho(a_2|h)(r_2^c + r(a_2, c)),$$

$$\rho(a_1|h)(r_1^c + r(a_1, d))\ \ +\ \ \rho(a_2|h)(r_2^d + r(a_2, d))\ \ \} \tag{2}$$

which is at least as high as his original payoff since $r_1^c \geq r_1^d$. (Note that the only difference between Eq. (1) and Eq. (2) is that the r_1^d term in the second line of Eq. (1) has been replaced by r_1^c in Eq. (2).)

Let $\{s_1, \ldots, s_k\}$ be the subset of pure strategies consistent with the history h. Let $\rho_1, \ldots, \rho_k$ be the probabilities associated with these strategies by Player 1's optimal mixed strategy ρ. Since T_1^c is consistent with $(\rho_1, \ldots, \rho_k)$, we can replace T_1^d by T_1^c without affecting these probabilities. Furthermore, the change to the strategies s_1 through s_k that corresponds to this replacement only affects outcomes that are consistent with h. Thus, the expected outcome at all other points in the tree remains the same and, therefore, the total expected payoff associated with the tree in which T_1^d is replaced by T_1^c is at least as high as the total expected payoff associated with the original tree. This same argument applies to T_2^d and T_2^c. Hence, Player 1 gains nothing by basing his strategies on Player 2's actions.

It is easy to see there is no reason to delay a fork since, for any tree which has a fork following a non-fork in a path, we can construct another tree which achieves the same expected payoff by simply shifting the fork up one level (Figure 4). Finally, once we have passed the last fork on a path, the strategy is uniquely determined by the history of play so far. Since Player 2 has access to this history and therefore knows all of Player 1's subsequent moves, Player 1 can do no better than to play one of his maxmin actions from that point onward. Furthermore, if he has more than one maxmin action, he can play the same maxmin action at the end of any path without affecting his expected payoff. ∎

The implications of Theorem 2.1 are twofold. First, Player 1 does not benefit from spending

Figure 4: The tree on the right achieves the same expected payoff as the tree on the left but with the fork on the leftmost path at stage 3 moved up to stage 2.

his resources trying to remember what Player 2 has done in the past. Second, he does not benefit from delaying the point at which two strategies differ, that is, he gains nothing by concealing his chosen strategy until some point later in the game.

3   Optimal mixed strategies when G is 2×2

Given that Player 1 has an optimal set with a simple tree representation, how can he find this set along with the associated optimal mixed strategy? We can describe a mixed strategy over the set by assigning a probability to each leaf of the corresponding tree since each leaf represents one of the strategies in the set. Alternatively, we could give a behavioral strategy description by assigning probabilities to the branches at each fork. The probability of a leaf is then just the product of the probabilities along the path from the root to the leaf. In general, many different probability vectors may be used throughout the tree. However, for 2×2 games, we show that the same probability vector can be used at every fork. Since this greatly simplifies the analysis, we begin by studying the case where the stage game is 2×2.

Proposition 3.1 shows that, when G is 2×2, there is an optimal set for G_K^N such that the corresponding optimal mixed strategy can be described by assigning the same probability vector to the branches at every fork and that this probability vector is the optimal mixed strategy for the stage game.

Proposition 3.1 *Let G be a 2×2 stage game in which $A_1 = \{a_1, a_2\}$. Let $(p, 1-p)$ be an optimal mixed strategy for Player 1 in G. For every $K \geq 1$, there is an optimal set S_K for G_K^N with associated optimal mixed strategy σ_K such that, at any fork in the simple tree representation of S_K, σ_K assigns probability p to the branch labeled a_1 and probability $(1-p)$ to the branch labeled a_2.*

Proof. Suppose we are given a simple tree representation for S_K which is an optimal set of pure strategies for G_K^N.

Case 1: The stage game has a saddle point.

In this case, by playing the same action at every stage, Player 1 can achieve an expected payoff of $V(G)$ in the repeated game. Thus, Player 1 has an optimal set of pure strategies for G_K^N containing a single element. This set can be represented by a tree consisting of a single node since it is understood that a stage game maxmin action is played at every stage after a leaf is reached.

Case 2: The stage game does not have a saddle point.

Let the payoff matrix for the stage game be $\begin{pmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \end{pmatrix}$. Consider Player 1's choice of action at a stage corresponding to a fork. There is an expected payoff V_1 for the remainder of the game given that he chooses action a_1 at this stage and an expected payoff V_2 given that he chooses a_2. Player 1's decision at any fork is then equivalent to the decision he faces in a 1-stage game with payoff matrix $\begin{pmatrix} r_{1,1} + V_1 & r_{1,2} + V_1 \\ r_{2,1} + V_2 & r_{2,2} + V_2 \end{pmatrix}$.

Suppose the modified matrix contains a saddle point. Assume without loss of generality that this saddle point occurs in row 2. Player 1 should then choose a_1 at this stage with probability 0. Thus no strategy starting with a_1 from this point is ever used. Let s_1 be any strategy in S_K

corresponding to a leaf in the subtree under the branch for a_1. Since s_1 is chosen with probability 0, $S_K \setminus \{s_1\}$ achieves the same value as S_K and is therefore optimal for G_K^N. Thus, we can create a new tree which represents another optimal set of pure strategies for G_K^N by removing the branch for a_1 along with the subtree underneath this branch.

We can assume, therefore, that the modified payoff matrix does not contain a saddle point. Adding a different constant to each row of a 2×2 matrix does not change the optimal mixed strategy for Player 1 as long as no saddle point is created or eliminated. Since there is no saddle point in either the original matrix or the modified matrix, the optimal mixed strategy must be the same for both matrices. Since S_K and σ_K are optimal, σ_K must assign probability p to the branch labeled a_1 and $1 - p$ the branch labeled a_2 at every fork. ∎

Proposition 3.1 implies that in a 2×2 game, Player 1, playing optimally, receives an expected payoff of $V(G)$ at each stage corresponding to a fork and r_* at every other stage. We can think of the probability of a particular fork occurring in a game as the product of the probabilities along the path from the root to the fork. The value of the repeated game in this case is:

$$V(G_K^N) \quad = \quad r_* + \frac{F(K,p)}{N}\left(V(G) - r_*\right) \tag{3}$$

where $F(K, p)$ is the maximum expected number of forks in a tree with K leaves in which every fork has two branches assigned probabilities according to the stage game optimal mixed strategy $(p, 1 - p)$.

Thus, for 2×2 games, we can compute S_K and σ_K by creating a tree with the maximum expected number of forks given that the actions at each fork are chosen according to $(p, 1 - p)$ and that no fork occurs at a depth $\geq N$. This is easily done by starting with one leaf assigned probability 1 and repeatedly expanding the most probable leaf having depth $< N$ according to $(p, 1 - p)$ until we have K leaves. The resulting tree represents an optimal set for G_K^N and the probabilities assigned to the leaves is the associated optimal mixed strategy.

3.1 Determining the value of G_K^N

To determine the value of G_K^N when G is 2×2, we need to determine the expected number of forks in such a tree. Our analysis makes use of the entropy function which measures the uncertainty in a random variable. It can also be viewed as measuring the uniformity of a probability distribution. (See Cover and Thomas (1991) for details on the properties of entropy described below.)

Definition 3.1 *Let X be a discrete random variable with probability function P. The* **entropy** *of X is defined to be*

$$H(X) \quad = \quad -\sum_x P(x) \log_2 P(x).$$

We also write the entropy of a probability vector $(p_1, p_2, \ldots, p_m)$ as

$$H(p_1, p_2, \ldots, p_m) \quad = \quad -\sum_{i=1}^m p_i \log_2 p_i$$

Joint entropy and conditional entropy are defined as follows.

Definition 3.2 *Let Y and Z be discrete random variables.*

- *The* **joint entropy** *$H(Y, Z)$ is the entropy of the joint distribution $P(y, z)$ defined by:*

$$H(Y, Z) = -\sum_{y \in Y} \sum_{z \in Z} P(y, z) \log P(y, z)$$

- *The* **conditional entropy** *$H(Z|Y)$ is defined by:*

$$H(Z|Y) = -\sum_{y \in Y} P(y) \sum_{z \in Z} P(z|y) \log P(z|y)$$

Conditional entropy measures the uncertainty of a random variable given that the value of another random variable is known. Joint and conditional entropy are related by the following property:

Proposition 3.2 (Chain Rule) $H(Y_1, \ldots, Y_n) = \sum_{i=1}^{n} H(Y_i | Y_{i-1}, \ldots, Y_1)$

Proposition 3.3 demonstrates how the expected number of forks is related to the entropy of the probability vector used at each fork and the entropy of the corresponding probability distribution over the leaves.

Proposition 3.3 *Assign a probability vector to each fork in a tree. This probability vector determines the probability of reaching each child of the fork given that the fork has been reached in a traversal of the tree. Let:*

- *$\{p^1, \ldots, p^m\}$ be the set of probability vectors assigned to the forks,*

- *x^i be the expected number of forks that are assigned probability vector p^i, $1 \leq i \leq m$, where again we take the probability of a fork occurring as the product of the probabilities along the path from the root to the fork,*

- *and ρ be the induced probability vector over the leaves in this tree.*

Then,

$$H(\rho) \;\; = \;\; \sum_{j=1}^{m} x^j H(p^j)$$

Proof. Let d be the maximum depth of a leaf in the tree. For each i, $0 \leq i \leq d$ define a random variable Y_i over the nodes at depth i such that for any $y \in Y_i$, $P(y) = Pr(Y = y)$ is the probability that y is reached when the tree is traversed according to the probabilities assigned to the edges. In order for the Y_i to be well defined we assume that any terminal node at depth $< d$ is extended in a straight line to depth d, i.e., each of the nodes in this extended subtree will have one child and we assign probability 1 to each edge in this subtree.

By the chain rule $H(Y_d, Y_{d-1}, \ldots, Y_0) = H(Y_{d-1} \ldots Y_0 | Y_d) + H(Y_d)$. But $H(Y_{d-1} \ldots Y_0 | Y_d) = 0$ since knowing which leaf has been reached uniquely determines the nodes reach at all other depths

15

in the tree. Therefore $H(Y_d) = H(Y_0, Y_1, \ldots, Y_d)$. Furthermore, for all i, $H(Y_i | Y_{i-1}, \ldots, Y_0) = H(Y_i | Y_{i-1})$ since, if we already know the parent of the node reached at depth i, knowing a distant ancestor of the node provides no additional information about which of the nodes at depth i has been reached.

By the definition of conditional entropy, for any $i > 0$, we have:

$$H(Y_i | Y_{i-1}) = \sum_{y \in Y_{i-1}} P(y) \sum_{z \in Y_i} P(z|y) \log P(z|y)$$

If z is not a child of y then $P(z|y) = 0$ and the inner sum is 0. If y is not a fork at depth $i - 1$ and z is y's child then $P(z|y) = 1$ so again the inner sum is 0. If y is a fork then the inner sum is $H(p^y)$ where p^y is the probability distribution used to expand y. Therefore, we have $H(Y_i | Y_{i-1}) = \sum_y P(y) H(p^y)$ where the sum is taken over all the forks y at depth $i - 1$. Let x_{i-1}^j be the expected number of forks at depth $i - 1$ that are assigned probability vector p^j, $1 \le j \le m$. Then $H(Y_i | Y_{i-1}) = \sum_{j=1}^m x_{i-1}^j H(p^j)$.

By the chain rule we have:

$$
\begin{aligned}
H(\rho) = H(Y_d) = H(Y_0, Y_1, \ldots, Y_d) &= \sum_{i=0}^d H(Y_i | Y_{i-1}, \ldots, Y_0) \\
&= \sum_{i=0}^d H(Y_i | Y_{i-1}) \\
&= \sum_{i=0}^d \sum_{j=1}^m x_{i-1}^j H(p^j) \\
&= \sum_{j=1}^m x^j H(p^j)
\end{aligned}
$$

$\blacksquare$

Corollary 3.1 *If every fork in a tree has m branches assigned probabilities $p_1, \ldots, p_m$, then the expected number of forks in the tree is:*

$$\frac{H(\rho)}{H(p_1, \ldots, p_m)}$$

where ρ is the induced probability vector over the leaves.

Proposition 3.1 and Corollary 3.1 imply the following result:

Theorem 3.1 *Fix $N \geq 1$ and $K \geq 2$. Let G be a 2×2 game where Player 1 has optimal mixed strategy $(p, 1-p)$. Let S_K be an optimal set for G_K^N having a simple tree representation. Let σ_K be the optimal mixed strategy for G_K^N induced by assigning $(p, 1-p)$ to each fork in the tree representing S_K. Then:*

$$V(G_K^N) \;\; = \;\; r_* + \frac{H(\sigma_K)}{H(p, 1-p)N} \left(V(G) - r_*\right) \tag{4}$$

Proof. According to Proposition 3.1, when G is 2×2, the branches at each fork will be assigned probabilities according to the stage game optimal mixed strategy $(p, 1-p)$. The result then follows from Corollary 3.1 since Player 1 receives an expected payoff equal to $V(G)$ at every fork. ∎

Neyman and Okada (1999) used a version of entropy to bound the value of repeated games in which one of the players is computationally restricted. They defined an extension to entropy called *strategic entropy* and use it to prove several results regarding repeated games with finite automata.

Definition 3.3 *(Neyman and Okada 1999) The* **N-strategic entropy**, *$H^N(\sigma)$, of a mixed strategy σ for Player 1 after N stages is defined by:*

$$H^N(\sigma) \;\; = \;\; \max_{t \in T} \left\{ -\sum_{\omega \in \Omega_{N+1}} P_{\sigma,t}(\omega) \log_2 P_{\sigma,t}(\omega) \right\}$$

where T is the set of pure strategies available to Player 2, Ω_N denotes the set of possible histories of length $N-1$ and $P_{\sigma,t}(\omega)$ is the probability of history ω given that Players 1 and 2 use strategies σ and t respectively.

Neyman and Okada (1999) show that if the strategic entropy of Player 1's mixed strategy is bounded by a function which is $o(N)$, then the value of the repeated game converges to Player 1's stage game maxmin value as N approaches infinity.

Since the strategies over which σ_K is defined are oblivious, $H(\sigma_K) = H_N^*(\sigma_K)$. Thus, Eq. (4) implies a direct relationship between strategic entropy and the value of the repeated game when Player 1 is restricted to mixing over a finite number of pure strategies and shows that strategic entropy arises naturally in the analysis of repeated games with finite strategy sets. Although originally used as a technical tool, Neyman and Okada (2000a) suggests that strategic entropy can be viewed as a "measure of the cost of randomization". The direct effect of entropy in Eq. (4) implies that entropy reflects the value of information rather than simply measuring the amount of information available to Player 1.

3.2 Bounding the Value of the Game

The only term in Eq. (4) that cannot be computed easily from the stage game is $H(\sigma_K)$. However, we can prove that $H(\sigma_K)$ is within an additive constant of $\log_2 K$ where the constant depends only on the stage game and is independent of K. Therefore, for large K we can replace $H(\sigma_K)$ in Eq. (4) by $\log_2 K$ without any significant loss of accuracy.

When $N \leq \frac{\log_2 K}{\log_2 |A_1|}$, Player 1 can create a complete tree of depth N having K leaves where each fork in the tree has $|A_1|$ branches. By assigning probabilities to each fork according to a stage game optimal mixed strategy, Player 1 can achieve an average expected payoff equal to $V(G)$ which is, therefore, the value of the repeated game in this case. Our work is focused on understanding the other extreme where N is so large compared to K that Player 1 can create an optimal tree with K leaves without any fork in the tree occurring at a depth greater than or equal to N. The value of the game in the intermediate case will obviously fall somewhere in between the values in the two extreme cases. The upper bound that we present is independent of the relationship between N and K while the lower bound requires N to be large enough that we do not need to be concerned with forks occurring beyond the end of the game. The following lemma provides a sufficient condition for this to be the case:

Lemma 3.1 *Let* $(p_1, \ldots, p_m)$ *be any probability vector such that* $p_i \geq p_{i+1}$ *for* $1 \leq i \leq m - 1$. *By starting with the root and repeatedly expanding the most probable leaf, create a tree with* K *leaves in which each fork has* m *branches assigned probabilities* $p_1, \ldots, p_m$. *The maximum depth of any leaf in the this tree is at most* $\lceil \frac{\log_2 K}{-\log_2 p_m} \rceil$.

Proof. Since there are K leaves, whenever a node has probability $\leq \frac{1}{K}$, it must be a leaf. Let x be the depth of any node. The probability of reaching that node is at most $(p_m)^x$. If $x \geq -\frac{\log_2 K}{\log_2 \hat{p}}$, then $(p_m)^x \leq \frac{1}{K}$ so this node must be a leaf. The maximum depth is therefore at most $\lceil -\frac{\log_2 K}{\log_2 p_m} \rceil$. ∎

Suppose Player 1 creates a tree using the same stage game optimal mixed strategy p at each fork. Corollary 3.1 indicates that we can find a lower bound for $V(G_K^N)$ by finding a lower bound for the entropy over the leaves in this tree. As we saw in Section 2, Player 1 can create this tree starting with one node and repeatedly expanding the most probable leaf using probability vector p for each expansion. In this process, each successive probability vector over the leaves should be more uniform than its predecessor since the largest probability is being split up into multiple elements with smaller probabilities. Intuitively, one would expect the entropy of this vector to be moving toward $\log_2 K$ or at least not moving away from it. In fact, as the following lemma shows, for given p, the difference between $\log_2 K$ and the entropy over the leaves is bounded above by a constant.

Lemma 3.2 *Let* $(p_1, \ldots, p_m)$ *be any probability vector such that* $p_i \geq p_{i+1}$ *for* $1 \leq i \leq m - 1$. *By starting with the root and repeatedly expanding the most probable leaf, create a tree with* K *leaves in which each fork has* m *branches assigned probabilities* $p_1, \ldots, p_m$. *Let* $q_1, \ldots, q_K$ *be the probability vector over the leaves in this tree. Then*

$$H(q_1, \ldots, q_K) \geq \log_2 K + \log_2 p_m$$

Proof. Let y be the probability of the last node that was expanded. Suppose $p_m y > q_i$ for some

19

$i, 1 \leq i \leq K$. Let z be the probability of leaf i's parent. Then $q_i \geq p_m z$ so $y > z$. But then the node with probability y should have been expanded before the node with probability z contradicting the assumption that y is the probability of the last node that was expanded. Thus $p_m y$ is the probability of the least probable leaf and is therefore $\leq \frac{1}{K}$.

Let $x = \max_{1 \leq i \leq K} q_i$. Since a node with probability y was expanded and there is a leaf with probability x, we have $x \leq y$. Therefore, $p_m x \leq p_m y \leq \frac{1}{K}$. So for $1 \leq i \leq K, q_i \leq x \leq \frac{1}{p_m K}$ which implies $\log_2 K + \log_2 p_m \leq -\log_2 q_i$. Consequently,

$$
\begin{aligned}
H(q_1, \ldots, q_K) &= \sum_{i=1}^{K} q_i (-\log_2 q_i) \\
&\geq \sum_{i=1}^{K} q_i (\log_2 K + \log_2 p_m) \\
&= \log_2 K + \log_2 p_m
\end{aligned}
$$

∎

In the 2×2 case, the expected number of forks in the simple tree representation of Player 1's optimal set of pure strategies is $\frac{H(\sigma_K)}{H(p, 1-p)}$ where $(p, 1 - p)$ is Player 1's stage game optimal mixed strategy. We know from information theory that $H(\sigma_K) \leq \log_2 K$. We can use this along with Lemma 3.2 to provide upper and lower bounds on $V(G_K^N)$, when G is 2×2, that are within an additive constant of optimal where this constant is independent of N and K and depends only on the stage game.

Theorem 3.2 *Let G be a 2×2 stage game. Let $(p, 1 - p)$ be Player 1's optimal mixed strategy for G and assume $p \geq 1 - p$. We have,*

$$
V(G_K^N) \leq r_* + \frac{\log_2 K}{H(p, 1-p)N}(V(G) - r_*) \tag{5}
$$

Furthermore, if $N \geq \lceil \frac{\log_2 K}{-\log_2 p} \rceil$ then

$$
V(G_K^N) \geq r_* + \frac{\log_2 K + \log_2 (1-p)}{H(p, 1-p)N}(V(G) - r_*) \tag{6}
$$

Proof. Since $H(\sigma_K) \leq \log_2 K$, Eq. (5) follows from Theorem 3.1. By Lemma 3.1, if $N \geq \lceil \frac{\log_2 K}{-\log_2 p} \rceil$ then the tree with K leaves having the maximum expected number of forks given that $(p, 1-p)$ is assigned to every fork contains no leaf at a depth $> N$. Hence, Eq. (6) is an immediate consequence of Lemma 3.2. ∎

Using a technique similar to the one used in Stearns (1989), Neyman and Okada (2000b) proved $V(G_K^N) \leq r_* + \frac{\log_2 K \max_{a,b} r(a,b)}{N}$. Eq. (5) is, in many cases, a tighter bound since it considers the actual payoffs received by Player 1 at each fork rather than simply using the fact that he gets at most $\max_{a,b} r(a,b)$ at each fork. Of course, both Eq. (5) and Neyman and Okada's bound imply that $\lim_{N \to \infty} V(G_K^N) = r_*$ if $\lim_{N \to \infty} \frac{\log_2 K}{N} = 0$. The bounds in Eq. (5) apply only to the case in which G is 2×2. In the next section, we develop similar bounds for the general case.

4 Approximating the optimal set when G is not 2×2

In the previous section, we showed that Player 1 has a very simple method for computing his optimal mixed strategy when the stage game is 2×2. Unfortunately, things are not so simple in the general case. As the following example shows, when the stage game is not 2×2, the probability vectors used on the forks in the simple tree representation of an optimal set are not always optimal mixed strategies for the stage game. Furthermore, different probability vectors may be required on different forks.

Example 4.1 *Consider a stage game with payoff matrix* $\begin{pmatrix} 0 & 9 \\ 6 & 5 \\ 5 & 6 \end{pmatrix}$. *The optimal mixed strategy is* $(0, 1/2, 1/2)$ *which results in an expected payoff of 5.5. A non-optimal mixed strategy,* $(1/10, 9/10, 0)$, *results in an expected payoff of 5.4. For $K = 3$ and $N = 2$, Figure 5 shows that these payoffs are close enough to make it beneficial to construct the tree with $(1/10, 9/10, 0)$ at the root rather than $(0, 1/2, 1/2)$. In this case, it is better for Player 1 to give his opponent more information about his current action in order to achieve an expected increase in the uncertainty*
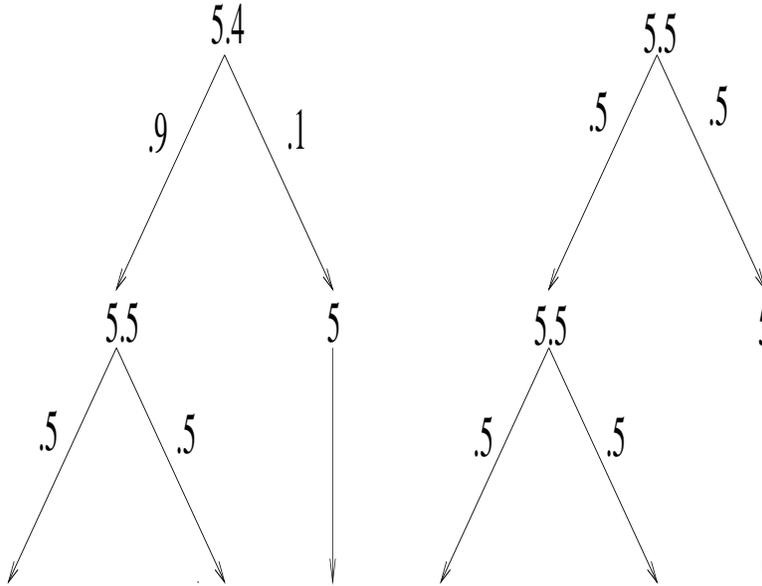
about his future actions.

Figure 5: The tree on the left achieves a total expected payoff of $5.4 + .9(5.5) + .1(5) = 10.85$ while the tree on the right achieves a total expected payoff of $5.5 + .5(5.5) + .5(5) = 10.75$. The numbers on the nodes indicate the expected payoff received at the stage when the node is reached.

In this section, we show that there is a probability vector p^* such that if Player 1 uses the procedure outlined in the previous section to create a tree with p^* assigned to each fork, then the induced mixed strategy achieves a total expected payoff that is within an additive constant of the optimal total expected payoff for G_K^N. This additive constant depends on p^* but is independent of N and K. Furthermore, p^* can be determined directly from the stage game.

In choosing a probability vector for a fork, Player 1 must consider both how the probability vector affects the expected payoff at that fork and how it affects the expected number of forks below that fork. We define the gain of a mixed strategy p as the expected amount above the pure maxmin payoff that the player receives by playing mixed strategy p in the one-stage game. Intuitively, it represents the effect that assigning p to a fork has on the expected payoff at that fork.

Definition 4.1 *The* **gain***, $g(p)$, of a mixed strategy p for Player 1 in a 1-stage game G is defined*

22

by:

$$g(p) = \left(\min_{a_2 \in A_2} \sum_{a_1 \in A_1} p(a_1) r(a_1, a_2) \right) - r_*$$

Similarly, the gain of a mixed strategy α for Player 1 in an N-stage game G^N is defined by:

$$g^N(\alpha) = \left(\min_{\tau \in \Delta(T)} E_{\alpha, \tau} \left[\sum_{n=1}^{N} r(a_1^n, a_2^n) \right] \right) - N r_*$$

We also need a way to measure the effect that assigning p to a fork has on the expected number of forks below that fork. Note that the expected number of forks in a subtree is the expected number of remaining stages in which Player 2 is uncertain about which pure strategy Player 1 is playing. Therefore, the less the expected number of forks below a fork, the more information Player 1's action at that fork reveals. Taking an information theoretic view of entropy, we can think of $H(p)$ as representing the average number of bits of information revealed by a single sample drawn according to probability vector p. Loosely equating these two notions of information, we compare mixed strategies based on their "gain per bit", $\frac{g(p)}{H(p)}$.

In the remainder of this section we use the following notation.

Notation 4.1 *Let $p = (p_1, \ldots, p_m)$ be any probability vector. By starting with the root and repeatedly expanding the most probable leaf, create a tree with K leaves in which each fork has m branches assigned probabilities $p_1, \ldots, p_m$. We use σ_K^p to denote the mixed strategy induced by this tree.*

We have the following upper and lower bounds on the total expected payoff that Player 1 can achieve by using a probability vector p at every fork:

Lemma 4.1 *Let $p = (p_1, \ldots, p_m)$ be any probability vector such that $p_i \geq p_{i+1}$ for $1 \leq i \leq m - 1$. Let $c \geq 0$ be the smallest integer such that $K - c = t(m - 1) + 1$ for some integer $t > 0$. Then,*

$$g^N(\sigma_K^p) \leq \frac{g(p)}{H(p)} \log_2 K$$

23

and if $N \geq \lceil \frac{\log_2 (K-c)}{-\log_2 p_1} \rceil$ *then*

$$g^N(\sigma_K^p) \geq \frac{g(p)}{H(p)} \left(\log_2 (K-c) + \log_2 p_m \right)$$

Proof. At each fork the gain achieved is $g(p)$ and at every non-fork the gain is 0. The lemma then follows from the same arguments used to prove Theorem 3.2. Note that since a tree in which every fork has m branches has $t(m-1)+1$ leaves for some $t > 0$, a tree created as described will have $K - c$ leaves. ∎

Using these bounds we can show that, by using the same probability vector at every fork, Player 1 can achieve a total expected payoff in G_K^N which is at most an additive constant below the optimal total expected payoff assuming N is sufficiently large with respect to K.

Theorem 4.1 *Fix N and K and let S_K be an optimal set for G_K^N having a simple tree representation. Let σ_K be the corresponding optimal mixed strategy. Let $\{p^1, \ldots, p^l\}$ be the set of probability vectors used on the forks in the simple tree representation of S_K. Assume $p^1 = (p_1^1, \ldots, p_m^1)$ has the maximum gain per bit among these vectors i.e. $\frac{g(p^1)}{H(p^1)} \geq \frac{g(p^i)}{H(p^i)}$ for $1 \leq i \leq l$. Then we have:*

$$
\begin{aligned}
g^N(\sigma_K) &= \frac{g(p^1)}{H(p^1)} H(\sigma_K) + \sum_{i=2}^{l} x^i \left(g(p^i) - \frac{g(p^1)}{H(p^1)} H(p^i) \right) \\
&\leq \frac{g(p^1)}{H(p^1)} \log_2 K
\end{aligned}
$$

Proof. The expected number of forks in this tree is $x^1 + \ldots + x^l$ where x^i is the expected number of forks which use probability vector p^i, $1 \leq i \leq l$. The optimal gain is:

$$g^N(\sigma_K) = x^1 g(p^1) + \ldots + x^l g(p^l)$$

From Proposition 3.3 we know:

$$x^1 H(p^1) + \ldots + x^l H(p^l) = H(\sigma_K)$$

which implies:

$$x^1 = \frac{H(\sigma_K) - x^2 H(p^2) - \ldots - x^l H(p^l)}{H(p^1)}$$

24

This leads to the following equation for the optimal gain:

$$g^N(\sigma_K) \quad = \quad \frac{g(p^1)}{H(p^1)}H(\sigma_K) + \sum_{i=2}^{l} x^i \left(g(p^i) - \frac{g(p^1)}{H(p^1)}H(p^i) \right)$$

Since $H(\sigma_K) \leq \log_2 K$ and we assumed $g(p^i) \leq \frac{g(p^1)}{H(p^1)}H(p^i)$ for all i, we have

$$g^N(\sigma_K) \leq \frac{g(p^1)}{H(p^1)} \log_2 K$$

∎

Let S_K be an optimal set for G_K^N and let $P(S_K)$ be the set of probability vectors assigned by the associated optimal mixed strategy to the forks in the simple tree representation of S_K. Theorem 4.1 implies that we could use the procedure outlined in the previous section with the probability vector with the maximum gain per bit in $P(S_K)$ to create an approximately optimal set. However, we would first have to determine $P(S_K)$. By using the results of Shapley and Snow (1950), we can compute a set $P^*(G)$, which depends only on G, such that $P^*(G) \supseteq P(S_K)$ for some optimal set S_K. By applying our procedure to the element of $P^*(G)$ with the maximum gain per bit, we can generate a set S'_K and a corresponding mixed strategy σ'_K such that the total expected payoff achieved by σ'_K is within an additive constant of the optimal.

Definition 4.2 *(Shapley and Snow, 1950) Let X and Y be the sets of optimal mixed strategies for Players 1 and 2 respectively in a game G. Let X^* and Y^* be the smallest sets whose convex hulls are X and Y respectively. A pair (x, y) is called a **solution** of G if $x \in X$ and $y \in Y$. A pair (x, y) is called a **basic solution** of G if $x \in X^*$ and $y \in Y^*$.*

Proposition 4.1 *(Shapley and Snow, 1950) A necessary and sufficient condition for a solution (x, y) of G be basic is that there exists a square sub-matrix M of G such that all three of the following equations hold:*

$$x^T \quad = \quad \frac{1^T(\text{adj } M)}{1^T(\text{adj } M)1}$$

$$y = \frac{(\text{adj } M)1}{1^T(\text{adj } M)1}$$

$$V(G) = \frac{\det M}{1^T(\text{adj } M)1}$$

where adj M *is the adjoint of matrix M,* det M *denotes M's determinant, 1 is the appropriately sized column vector consisting entirely of 1's and the condition is considered not to hold when the right hand terms are indeterminate.*

Definition 4.3 *If $x \in X^*$, where X^* is as defined in Definition 4.2, then x is called a* **basic mixed strategy** *for Player 1.*

Definition 4.4 *The set of* **potentially basic mixed strategies** *for Player 1 is the set, $P^*(G)$, of mixed strategies, x, such that $x^T = \frac{1^T(\text{adj } M)}{1^T(\text{adj } M)1}$ for some square sub-matrix M of G.*

Note that, for a mixed strategy to be potentially basic, only the first of the three equations from Proposition 4.1 is required to hold. Hence, a potentially basic mixed strategy is not necessarily a basic mixed strategy.

To show that there is an optimal set S_K which uses only mixed strategies from $P^*(G)$ on its forks, we need the following property of the adjoint of a matrix:

Lemma 4.2 *Let A be a square matrix and let A' be created from A by adding a different constant to each row i.e. $A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$ and $A' = \begin{pmatrix} a_{11} + c_1 & \dots & a_{1n} + c_1 \\ \vdots & \ddots & \vdots \\ a_{n1} + c_n & \dots & a_{nn} + c_n \end{pmatrix}$. Then $1^T\text{adj } A = 1^T\text{adj } A'$.*

Proof. It suffices to show that $1^T\text{adj } A = 1^T\text{adj } A'$ when only one of the c_i's say c_l is nonzero for some $l, 1 \leq l \leq n$. Let $A(j|i)$ be the matrix formed by removing row j and column i from A. We need to show

$$\sum_{i=1}^{m}(-1)^{i+j}\det A(j|i) = \sum_{i=1}^{m}(-1)^{i+j}\det A'(j|i) \tag{7}$$

for every $j, 1 \leq j \leq n$.

When $j = l$, Eq. (7) holds since $A(l|i)$ is identical to $A'(l|i)$ for all $i, 1 \leq i \leq n$. By the linearity of the determinant (see Hoffman and Kunze, 1971, p. 142), $\det A'(j|i) = \det A(j|i) + \det B(j|i)$ where

$$B = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{(l-1)1} & \cdots & a_{(l-1)n} \\ c_l & \cdots & c_l \\ a_{(l+1)1} & \cdots & a_{(l+1)n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

Thus we need to show that $\sum_{i=1}^{n}(-1)^{i+j} \det B(j|i) = 0$ for $j \neq l$. Let $\beta_{jk} = (-1)^{j+k} \det B(j|k)$. We know $b_{i1}\beta_{j1} + \ldots + b_{in}\beta_{jn} = 0$ for each i and j such that $1 \leq j \neq i \leq n$ (see Agnew and Knapp, 1983, Theorem 8, p. 111). In particular, since $b_{lk} = c_l$ for each $k, 1 \leq k \leq n$, we have $\beta_{j1} + \ldots + \beta_{jn} = 0, 1 \leq j \neq l \leq n$ which implies Eq. (7) for every $j, 1 \leq j \neq l \leq n$. Therefore, $1^T \text{adj } A = 1^T \text{adj } A'$. ∎

The set of potentially basic mixed strategies for Player 1 in G contains the set of mixed strategies used on the forks in the tree representation of some optimal set S_K for G_K^N.

Theorem 4.2 *Fix N and K. Let S_K be an optimal set for G_K^N that has a simple tree representation. By assigning only potentially basic mixed strategies to the forks in this tree, we can induce an optimal mixed strategy σ_K for G_K^N.*

Proof. Consider any fork with l branches in the tree representation of S_K. Let V_i be the expected payoff associated with the the subtree under branch $i, 1 \leq i \leq l$. Player 1's decision at this fork is then equivalent to the decision he faces in a 1-stage game defined by

$$M = \begin{pmatrix} r_{1,1} + V_1 & \cdots & r_{1,n} + V_1 \\ \vdots & \ddots & \vdots \\ r_{l_1} + V_l & \cdots & r_{l,n} + V_l \end{pmatrix}$$

27

As in the proof of Proposition 3.1, we can assume that M does not have a saddle point. It is sufficient for Player 1 to choose a basic mixed strategy, x, for M. By Proposition 4.1, x must satisfy $x^T = \frac{1^T(\text{adj } M')}{1^T(\text{adj } M')1}$ for some square sub-matrix M' of M. By Lemma 4.2, $1^T(\text{adj } M') = 1^T(\text{adj } M'')$ where M'' is formed from M' by subtracting the V_i's from their corresponding rows. M'' is therefore a square sub-matrix of the payoff matrix for the stage game G which implies that x is a potentially basic mixed strategy. ∎

Using Theorem 4.2, we can develop upper and lower bounds on $g^N(\sigma_K)$ based on the potentially basic mixed strategy with the maximum gain per bit.

Theorem 4.3 *Let $p^* = (p_1^*, \ldots, p_m^*)$ be the potentially basic mixed strategy with the maximum gain per bit i.e. $p^* = \arg\max_{p \in P^*(G)} \frac{g(p)}{H(p)}$. Assume $p_i^* \geq p_{i+1}^*$ for $1 \leq i \leq m-1$. Let σ_K be an optimal mixed strategy for G_K^N. Let $c \geq 0$ be the smallest integer such that $K - c = t(m-1) + 1$ for some integer $t > 0$. For any $N \geq 1$,*

$$g^N(\sigma_K) \;\leq\; \frac{g(p^*)}{H(p^*)} \frac{\log_2 K}{N} \tag{8}$$

Furthermore, if $N \geq \lceil \frac{\log_2(K-c)}{-\log_2 p_1^*} \rceil$ then

$$g^N(\sigma_K) \geq \frac{g(p^*)}{H(p^*)} \frac{\log_2(K-c) + \log_2 p_m^*}{N} \tag{9}$$

Proof. Fix N and K. Let S_K be an optimal set for G_K^N having a simple tree representation T By Theorem 4.2, we can create an optimal mixed strategy σ_K by assigning only potentially basic mixed strategies to the forks in T. Let $\{p^1, \ldots, p^l\}$ be the set of potentially basic mixed strategies used to construct σ_K. Assume the p^i are ordered in decreasing order of their gain per bit, i.e $\frac{g(p^i)}{H(p^i)} \geq \frac{g(p^{i+1})}{H(p^{i+1})}$ for $1 \leq i < l$. By Theorem 4.1, we have

$$g^N(\sigma_K) \leq \frac{g(p^1)}{H(p^1)} \frac{\log_2 K}{N} \leq \frac{g(p^*)}{H(p^*)} \frac{\log_2 K}{N}$$

since $\frac{g(p^*)}{H(p^*)} \geq \frac{g(p^1)}{H(p^1)}$.

Eq. (9) follows directly from Lemma 4.1. ∎

Corollary 4.1 *Let p^* and c be as in Theorem 4.3. For any $N \geq 1$,*

$$V(G_K^N) \leq r_* + \frac{g(p^*)}{H(p^*)} \frac{\log_2 K}{N}$$

Furthermore, if $N \geq \lceil \frac{\log_2 (K-c)}{-\log_2 p_1^*} \rceil$ then

$$V(G_K^N) \geq r_* + \frac{g(p^*)}{H(p^*)} \frac{\log_2 (K - c) + \log_2 p_m^*}{N}$$

Proof. This follows immediately from Theorem 4.3. ∎

For sufficiently large K, $c \leq K/2$. Therefore, using the potentially basic mixed strategy p^* with the maximum gain per bit, Player 1 can achieve a total expected payoff within $\frac{g(p^*)}{H(p^*)} (\log_2 p_m^* - 1)$ of the optimal. Since potentially basic mixed strategies are the only probability vectors that need to be considered for any fork, there is no other single probability vector that Player 1 can use to achieve a total expected payoff within this constant of the optimal for every K. In other words, when K is sufficiently large, p^* is the best probability vector that Player 1 can use if he creates his tree using the same mixed strategy at every fork.

Experiments indicate that the total expected payoff is closer to the upper bound than the lower bound which suggests that the upper bound is probably a better estimate of the total expected payoff achieved by the approximately optimal strategy. For example consider the following game:

$$\begin{pmatrix} 3 & 1 & 5 \\ 0 & 0 & 9 \\ 5 & 1 & 3 \\ 2 & 4 & 1 \end{pmatrix}$$

There are eight potentially basic mixed strategies to consider. These are shown in Tables I and II along with their gain, entropy, and gain per bit.

Several other potentially basic mixed strategies were eliminated from consideration because they did not have a positive gain or because their entropy was the same as another strategy but

Table I: Comparison of eight potentially basic mixed strategies for $K = 100$.

	p	$g(p)$	$H(p)$	$\dfrac{g(p)}{H(p)}$	UB_{100}	$g^N(\sigma^p_{100})$	LB_{100}
1	$\left(0, \frac{1}{10}, 0, \frac{9}{10}\right)$	0.80	0.469	1.706	11.33	10.48	5.67
2	$\left(\frac{1}{2}, 0, 0, \frac{1}{2}\right)$	1.50	1.000	1.500	9.97	9.84	8.47
3	$\left(\frac{1}{3}, 0, 0, \frac{2}{3}\right)$	1.33	0.981	1.452	9.65	9.54	7.35
4	$\left(\frac{3}{7}, 0, 0, \frac{4}{7}\right)$	1.43	0.985	1.450	9.63	9.57	7.86
5	$\left(\frac{4}{11}, 0, \frac{1}{11}, \frac{6}{11}\right)$	1.64	1.322	1.238	8.22	7.87	3.92
6	$\left(0, 0, \frac{3}{5}, \frac{2}{5}\right)$	1.20	0.971	1.236	8.21	8.15	6.57
7	$\left(0, \frac{4}{31}, \frac{9}{31}, \frac{18}{31}\right)$	1.61	1.355	1.191	7.91	7.65	4.37
8	$\left(0, \frac{1}{4}, 0, \frac{3}{4}\right)$	0.50	0.811	0.616	4.09	4.01	2.86

Table II: Comparison of eight potentially basic mixed strategies for $K = 10^6$.

	p	$g(p)$	$H(p)$	$\dfrac{g(p)}{H(p)}$	UB_{10^6}	$g^N(\sigma^p_{10^6})$	LB_{10^6}
1	$\left(0, \frac{1}{10}, 0, \frac{9}{10}\right)$	0.80	0.469	1.706	34.00	33.28	28.33
2	$\left(\frac{1}{2}, 0, 0, \frac{1}{2}\right)$	1.50	1.000	1.500	29.90	29.86	28.40
3	$\left(\frac{1}{3}, 0, 0, \frac{2}{3}\right)$	1.33	0.981	1.452	28.94	28.83	26.63
4	$\left(\frac{3}{7}, 0, 0, \frac{4}{7}\right)$	1.43	0.985	1.450	28.90	28.84	27.12
5	$\left(\frac{4}{11}, 0, \frac{1}{11}, \frac{6}{11}\right)$	1.64	1.322	1.238	24.67	24.34	20.39
6	$\left(0, 0, \frac{3}{5}, \frac{2}{5}\right)$	1.20	0.971	1.236	24.63	24.58	23.00
7	$\left(0, \frac{4}{31}, \frac{9}{31}, \frac{18}{31}\right)$	1.61	1.355	1.191	23.73	23.49	20.21
8	$\left(0, \frac{1}{4}, 0, \frac{3}{4}\right)$	0.50	0.811	0.616	12.28	12.20	11.05

their gain was lower. The strategies are listed in decreasing order of their gain per bit. In the tables, $\text{UB}_K = \frac{g(p)}{H(p)} \log_2 K$ is the upper bound on the gain given that p is used at every fork. $\text{LB}_K = \frac{g(p)}{H(p)} \left(\log_2 (K - c) + \log_2 p_m \right)$ is the lower bound on the gain given that p is used at every fork where c is as defined in Theorem 4.3 and p_m is the smallest element in the support of p. The upper bound, lower bound, and actual gain $g^N(\sigma_K^p)$ achieved by using p at every fork are shown for $K = 100$ in Table I and $K = 1,000,000$ in Table II. Note that, since the strategies are ordered by their gain per bit, they are ordered by their upper bounds. Had we chosen a large enough K, they would also have been ordered by LB_K. In fact, for large enough K, the lower bound for strategy i will be larger than the upper bound for strategy $i + 1$.

The optimal mixed strategy is $(4/11, 0, 1/11, 6/11)$ which is strategy number 5 in the tables. Strategies 7 and 8 will not be used in an optimal tree since there are other potentially basic mixed strategies with the same size support which have higher gain and lower entropy. The upper bound is a fairly accurate measure of the true gain for all of the strategies and is closer than the lower bound to the actual gain in each case. Strategy 1 has the largest gain per bit at 1.706. It is already significantly superior to the other strategies when $K = 100$ even though the lower bound for Strategy 1 is well below the lower bounds for other strategies in the list at this point. The optimal gain for G_{100}^N was computed to be 10.98 which is only 0.5 above the gain achieved by our approximately optimal strategy. The value of $G_{10^6}^N$ was not computed since the dynamic programming algorithm to compute the value of the game exactly would require an enormous amount of time for such a large value for K. This demonstrates the benefit of having such a simple procedure for computing an approximately optimal mixed strategy.

5 Relationship to Strategic Entropy

In this section, we discuss in more detail the relationship between our results and Neyman and Okada's results on repeated games in which Player 1 has bounded strategic entropy. Recall that strategic entropy is defined as follows:

Definition 5.1 *(Neyman and Okada 2000a) Given $\sigma \in \Delta(S)$ and $t \in T$, let $(X_k)_{k=1}^{\infty}$ be the random play induced by (σ, t). Define $H^N(\sigma : t)$ by:*

$$H^N(\sigma : t) = H(X_1, \ldots, X_N) = - \sum_{\omega \in \Omega_{N+1}} P_{\sigma,t}(\omega) \log_2 P_{\sigma,t}(\omega)$$

where Ω_N denotes the set of possible histories of length $N - 1$ and $P_{\sigma,t}(\omega)$ is the probability of history ω given that Players 1 and 2 use strategies σ and t respectively.

The **N-strategic entropy**, *$H^N(\sigma)$, of σ is defined by:*

$$H^N(\sigma) = \max_{t \in T} H^N(\sigma : t)$$

For any $\eta \geq 0$ define $\Sigma^N(\eta) = \{\sigma \in \Delta(S) : H^N(\sigma) \leq \eta\}$. Let $G^N(\eta)$ be the N-stage repetition of G in which Player 1 is restricted to mixed strategies in $\Sigma^N(\eta)$ while Player 2 remains unrestricted. In other words, $G^N(\eta)$ is the N-stage repetition of G in which Player 1 is restricted to mixed strategies with entropy at most η. Neyman and Okada (2000a) point out that $\Sigma^N(\eta)$ may not be convex and therefore $G^N(\eta)$ may not have a value. Thus they provide bounds on $W^N(\eta) = \max_{\sigma \in \Sigma^N(\eta)} \min_{t \in T} r^N(\sigma, t)$ where $r^N(\sigma, t) = E_{\sigma,t}\{\frac{1}{N} \sum_{i=1}^{N} r(a_i, b_i)\}$. The bounds on $W^N(\eta)$ depend on the following function:

Definition 5.2 *For any $\gamma \geq 0$, define $U(\gamma) = \max_{\substack{\alpha \in \Delta(A) \\ H(\alpha) \leq \gamma}} \min_{b \in B} r(\alpha, b)$.*

$U(\gamma)$ is the maximum gain achieved in the stage game G by a strategy with entropy at most γ. U is strictly increasing and piecewise convex up to the point $\bar{\gamma} = \min\{H(p) : p$ is an optimal mixed strategy for stage game $G\}$ after which it is a constant $V(G) - r_*$. See Figure 6.

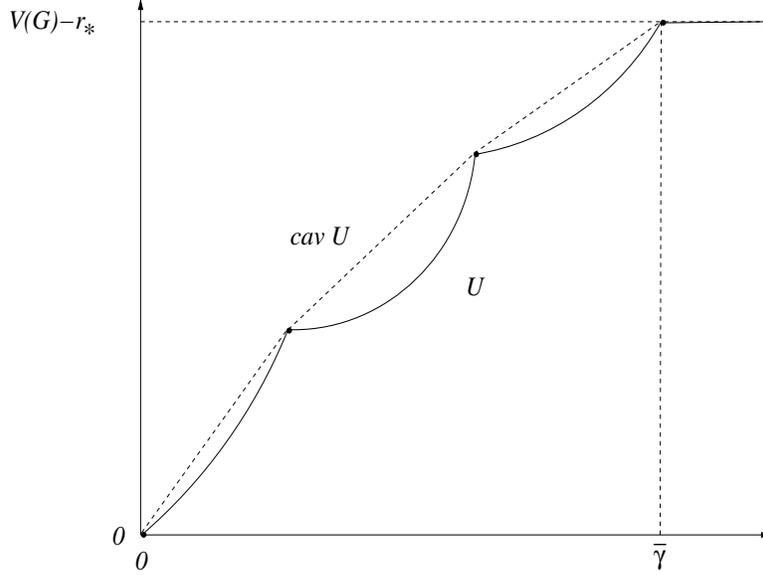Neyman and Okada (2000a) proved the following bounds on $W^N(\eta)$:

Figure 6: An example of cav U.

Proposition 5.1 *(Neyman and Okada, 2000a) For every N,*

$$(\text{cav } U)(\frac{\eta}{N}) - \frac{V(G) - r_*}{N} \leq W^N(\eta) \leq (\text{cav } U)(\frac{\eta}{N})$$

where cav U *is the concavification of U.*

Since Player 1's mixed strategies in G_K^N have at most K pure strategies in their support, it must be the case that:

$$V(G_K^N) \leq W^N(\log_2 K) \leq (\text{cav } U)\left(\frac{\log_2 K}{N}\right).$$

One can show that $(\text{cav } U)(\frac{\eta}{N}) \leq \frac{g(p^*)}{H(p^*)}\frac{\log_2 K}{N}$ so Neyman and Okada's upper bound is tighter than the one given in Theorem 4.3. However, as we now demonstrate, when $\frac{\log_2 K}{N} \leq H(p^*)$, $(\text{cav } U)(\frac{\log_2 K}{N}) = \frac{g(p^*)}{H(p^*)}\frac{\log_2 K}{N}$. (See Figure 7.)

Lemma 5.1 *For all $p \in \Sigma(A_1)$, $\frac{g(p)}{H(p)} \leq \frac{g(p^*)}{H(p^*)}$.*

Proof. By Lemma 4.1, we know $g^N(\sigma_K^p) \geq \frac{g(p)}{H(p)}[\log_2(K - c) + \log_2 p_m]$ when $N \geq \frac{\log_2 K}{-\log_2 p_1}$. If $\frac{g(p)}{H(p)} > \frac{g(p^*)}{H(p^*)}$ then, for sufficiently large K and N,

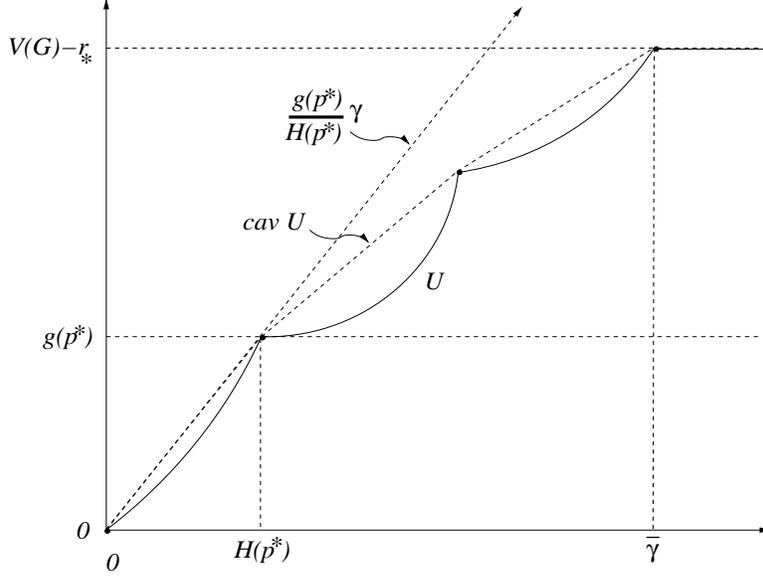$$g^N(\sigma_K^p) \geq \frac{g(p^*)}{H(p^*)} \log_2 K.$$

33

Figure 7: cav U and $\frac{g(p^*)}{H(p^*)}$.

But this is impossible since $\frac{g(p^*)}{H(p^*)} \log_2 K \geq g^N(\sigma_K) \geq g^N(\sigma_K^p)$ for all p, N, and K. ∎

Lemma 5.2 $g(p^*) = U(H(p^*))$

Proof. Suppose not. Then there is a p such that $g(p) > g(p^*)$ and $H(p) \leq H(p^*)$. But then $\frac{g(p)}{H(p)} > \frac{g(p^*)}{H(p^*)}$. This is impossible by Lemma 5.1. ∎

Proposition 5.2 *When* $\gamma \leq H(p^*)$, $\frac{g(p^*)}{H(p^*)}\gamma = (\text{cav } U)(\gamma)$.

Proof. By Lemma 5.1, no point on $U(\gamma)$ can be above the line from the origin to $(H(p^*), g(p^*))$ since the slope of this line is $\frac{g(p^*)}{H(p^*)}$. Thus $(\text{cav } U)(\gamma) \leq \frac{g(p^*)}{H(p^*)}\gamma$ for all γ. Furthermore, since $U(0) = 0$ and $U(H(p^*)) = g(p^*)$ the line connecting the origin to $(H(p^*), g(p^*))$ must be below $(\text{cav } U)(\gamma)$ when $\gamma \in [0, H(p^*)]$. Therefore, $\frac{g(p^*)}{H(p^*)}\gamma \leq (\text{cav } U)(\gamma)$ for $\gamma \in [0, H(p^*)]$. ∎

While it is easy to apply Proposition 5.1 to obtain an upper bound on $V(G_K^N)$, using it to obtain a lower bound is somewhat problematic. It is impossible for $\Sigma(\eta)$ to be a subset of the mixed strategies available to Player 1 in G_K^N since for any η, we can construct a mixed strategy with more than K pure strategies in its support but having entropy at most η. Whether such

34

a mixed strategy would yield a higher payoff that $V(G_K^N)$ would depend on the payoff matrix. The problem then becomes one of determining a value for η such that $W^N(\eta) \leq V(G_K^N)$. As the following example illustrates, if such an η exists it may be very low compared to $H(\sigma_K)$.

Consider the following variation of the game in Example 4.1:

$$\begin{pmatrix} 0 & 999 \\ 6 & 5 \\ 5 & 6 \end{pmatrix}$$

The optimal strategy is $(0, 1/2, 1/2)$ yielding a payoff of 5.5. There is a much less uniform sub-optimal strategy $(0.001, 0.999, 0)$ that achieves a payoff of 5.4. Suppose $K = 3$. The optimal mixed strategy σ_3 is constructed from a tree with three leaves where $(.999, .001)$ is assigned to the root and $(1/2, 1/2)$ assigned to he left most child of the root, i.e., the child under .999. We have $g^N(\sigma_3) = 0.8995$ and $H(\sigma_3) = 1.01$. However, if we constructed a tree with three leaves using $(0.999, 0.001)$ at every fork then the entropy of the corresponding mixed strategy would be 0.02. The entropy of the mixed strategy corresponding to a tree with four leaves in which $(0.999, 0.001)$ is assigned to every fork would be 0.03. The gain of this mixed strategy would be 1.20 which is larger than $g^N(\sigma_K)$. Therefore, we would have to set η below 0.03 for $W^N(\eta)$ to be a lower bound on $V(G_3^N)$. We suspect that any general lower bound on $V(G_K^N)$ will depend much more heavily than Proposition 5.1 on the details of the payoff matrix as was the case in Theorem 4.3. If Proposition 5.1 could be used to establish a general lower bound on $V(G_K^N)$, however, it may enable us to analyze the intermediate case in which $N \leq \frac{\log_2 K}{-\log_2 p_1^*}$.

6   Computational Models Revisited

In the introduction, we explained that we would like to measure the computational resources available to Player 1 in terms of the amount of memory required to implement a pure strategy using a computer program rather than the number of states required to implement a pure strategy on a finite automaton. However, we would also like to restrict the player to use a small constant

amount of time at each stage. Such a model would provide us with a measure of a strategy's complexity that is an accurate measure of the strategy's hardware requirements without sacrificing the time characteristics of a finite automaton.

Let G_m^N be the N-stage repetition of G in which Player 1 is restricted to an m-bit memory and is further restricted to using a constant amount of time at each stage. Since he can initialize this memory in 2^m ways, Player 1 can play at most 2^m distinct pure strategies on this machine. Therefore, the value of G_m^N is at most $V(G_K^N)$ where $K = 2^m$. Hence, Theorem 4.3 implies $V(G_m^N) \leq r_* + \frac{g(p^*)}{H(p^*)} \frac{m}{N}$.

We can use the results of the previous section to develop a lower bound as well. Let S_K^* be the set of K pure strategies created by starting at the root and repeatedly expanding the most probable fork using the potentially basic mixed strategy p^* with the maximum gain per bit. Since each pure strategy in S_K^* is oblivious, we can uniquely describe any pure strategy in S_K^* by simply listing the actions that the strategy takes at each stage. To implement a pure strategy from S_K^*, Player 1 needs to store the strategy and the default maxmin action to be played when the strategy no longer specifies an action. It then needs to move through the strategy's description one action at a time playing the action specified until the strategy's description ends. It plays the default action from that point onward. This can certainly be done in a small constant amount of time at each stage with a small constant amount of additional memory needed to store the default action.

Lemma 3.1 implies that each strategy in S_K^* specifies at most $\frac{\log_2 K}{-\log p_1^*}$ actions before defaulting to a maxmin action for the remainder of the game where again p_1^* is the highest probability in the support of p^*. Let $|p^*|$ be the size of the support of p^*. It takes $\lceil \log_2 |p^*| \rceil$ bits to write down a single action so the complete description of a pure strategy would require at most $\frac{\log_2 K \lceil \log_2 |p^*| \rceil}{-\log_2 p_1^*}$ bits. If Player 1 has m bits of memory then he can describe all the pure strategies in S_K where

$\log_2 K = \frac{m \log_2 (1/p_1^*)}{\lceil \log_2 |p^*| \rceil}$. Theorem 4.3 then implies that, if $N \geq \lceil \frac{m}{-\log_2 p_1^*} \rceil$,

$$V(G_m^N) \quad \geq \quad r_* + \frac{g(p^*)}{H(p^*)} \frac{\log_2 \frac{1}{p_1^*}}{\lceil \log_2 |p^*| \rceil} \frac{m}{N} - \frac{c'}{N}$$

where c' is a constant that is independent of N and m. As p_1^* grows, this lower bound decreases so different encoding schemes may be more useful when p_1^* is large. For example, in O'Connell (2000) we provide an encoding that describes K pure strategies using at most $5.5 \log_2 K$ bits for the case when $|p^*| = 2$ regardless of the magnitude of p_1^*.

Suppose Player 1 is restricted to an m-bit memory and is using the simple encoding scheme described above. If he has n actions to chose from in the stage game then, if he created a set of pure strategies that corresponds to a complete n-ary tree of depth m, he could describe every strategy in this set using exactly $m \lceil \log_2 n \rceil$ bits. This is true since every strategy would specify m actions and every action would take $\lceil \log_2 n \rceil$ bits to describe. If any of the strategies specified less than m actions, he would be not be fully utilizing his available memory when he was playing that strategy since some bits in memory would be uninitialized. Therefore, the optimal set of pure strategies for this encoding scheme is described by a complete tree. The question is how many branches should a fork have and what probabilities should be assigned to those branches. Suppose we again confine ourselves to trees that use the same probability vector at each fork. By the argument used in the proof of Theorem 4.2, we need only consider potentially basic mixed strategies. However, it may not be optimal to use p^* since we no longer need to be concerned about future uncertainty created by our choice of probabilities at each stage. This is because every leaf in the tree now has the same depth. The expected number of forks in this tree is $\frac{m}{\lceil \log_2 |p| \rceil}$ since each level in the tree uses up $\lceil \log_2 |p| \rceil$ of the m bits. The total gain achieved by this mixed strategy is, therefore, $\frac{g(p)}{\lceil \log_2 |p| \rceil} m$. Thus, the best choice for p is the potentially basic mixed strategy that maximizes $\frac{g(p)}{\lceil \log_2 |p| \rceil}$. This gives us the following result:

Theorem 6.1 *Let G_m^N be the N-stage repetition of G in which Player 1 is restricted to an m-bit memory and is further restricted to using a constant amount of time at each stage. Let $P^*(G)$ be the set of potentially basic mixed strategies for G. Let $\hat{p} = \arg\max\limits_{p \in P^*(G)} \dfrac{g(p)}{\log_2 |p|}$ and let $p^* = \arg\max\limits_{p \in P^*(G)} \dfrac{g(p)}{H(p)}$. For all N and m such that $N \geq \dfrac{m}{\lceil \log_2 |p| \rceil}$.*

$$r_* + \left\lfloor \frac{m - c}{\lceil \log_2 |\hat{p}| \rceil} \right\rfloor \frac{g(\hat{p})}{N} \leq V(G_m^N) \leq r_* + \frac{g(p^*)}{H(p^*)} \frac{m}{N}$$

where c represents the small constant amount of memory required to store the program that determines the next action based on the strategy's description.

Proof. The result follows immediately from Theorem 4.3 and the above discussion. ∎

Returning to our example from Section 4, Table I shows the eight potentially basic mixed strategies. In this example, $\hat{p} = (1/2, 0, 0, 1/2)$ while $p^* = (0, 1/10, 0, 9/10)$ is one of the worst probability vectors to use when creating a complete tree. $V(G_m^N)$ is between $1.5m/N$ and $1.706m/N$.

Although most work in bounded rationality considers only the computational resources necessary to execute a player's strategy, Papadimitriou (1992) suggests that the time need to find an optimal strategy is also important. In O'Connell (2000), we show that given a probability vector p, Player 1 can construct σ_K^p in $O(K)$ time. This means that to construct an optimal mixed strategy for G_K^N when G is 2×2, Player 1 first needs to compute the optimal mixed strategy for the stage game. In the general case, Player 1 can construct an approximately optimal mixed strategy $\sigma_K^{p^*}$ by first computing the potentially basic mixed strategy p^* with the maximum gain per bit. In either case, the additional computation depends only on the stage game and is independent of K and N. Therefore, the time needed to construct an optimal mixed strategy in the 2×2 case and an approximately optimal mixed strategy in the general case is $O(K)$.

7   Conclusions

We have argued that the number of states needed to execute a strategy on a finite automaton is not a particularly good measure of the strategy's complexity. It would be preferable to measure complexity in terms of memory required to implement the strategy using a computer program since this is a better measure of the strategy's physical resource requirements. This is the approach taken in Stearns (1989). Unfortunately, the memory measure ignores the time needed by the player at each stage to determine his action. The state counting measure's main (and perhaps only) advantage over the memory measure is that the time that a finite automaton takes at each stage is considered to be constant. By studying repeated games in which one of the players is restricted to some finite number of pure strategies, we have developed results that are independent of the computational model on which those strategies are executed. Using these results, we developed fairly tight bounds on the value of repeated games in which one of the players uses a general computing device with a limited memory but is further restricted to using a constant amount of time at each stage. Thus we are able to combine the best of both measures – using memory as a measure of a strategy's complexity while simultaneously restricting the time used by the player's computer to be similar to the time requirements of a finite automaton. In doing so, we also see that such a computational restriction results in strategies that are consistent with the intuition behind Theorem 2.1 – Player 1 has no reason to base his actions on Player 2's previous actions since they do not provide any information about Player 2's future play. On the other hand, when Player 1 is restricted to an m-state finite automaton, his optimal mixed strategy will no doubt include non-oblivious strategies in its support. The reason for this is that there are only $O(2^m)$ oblivious m-state finite automata while there are exponentially more m-state finite automata that base their actions on the opponent's past actions. When restricted to finite automata, Player 1 bases his play on Player 2's past actions not because these actions provide any useful information

but because doing so increases the number of pure strategies available to Player 1. This behavior is an artifact of the computational model rather than something inherent in repeated games in which one of the players is computationally restricted.

Our analysis also adds insight into the connection between strategic entropy and the value of repeated games with finite strategy sets. Far from simply being a technical tool, entropy arises naturally in the analysis of such games and, therefore, repeated games in which one of the players is computationally restricted. Entropy appears as a factor in an exact formula for the value of the game and thus is seen to have a direct numerical effect on the game's value.

Finally, our approach does not readily provide a way to analyze games in which both players are bounded. If we restrict both players to choosing sets of size K, the best set for Player 1 depends on the set that Player 2 chooses and vice versa. If we allow the choice of sets to be randomized, then any restriction on the players caused by limiting them to sets of size K vanishes. This is the case because a mixed strategy in the unrestricted repeated game is a randomized choice over sets of size one. There does not appear to be any other obvious way to model this type of game.

References

Agnew, J. and R. C. Knapp (1983). *Linear Algebra with Applications*. Brooks/Cole Publishing Co., Monterey, CA.

Aumann, R. (1997). Rationality and bounded rationality. *Games Econ. Behavior 21*, 2–14, doi:10.1006/game.1997.0585.

Ben-Porath, E. (1993). Repeated games with finite automata. *J. of Econ. Theory 59*, 17–32, doi:10.1006/jeth.1993.1002.

Cover, T. and J. Thomas (1991). *Elements of Information Theory*. John Wiley and Sons, Inc., New York, NY.

Hoffman, K. and R. Kunze (1971). *Linear Algebra*. Prentice-Hall, Inc., Englewood Cliffs, NJ.

Hopcroft, J. and J. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA.

Kalai, E. (1993). Bounded rationality and strategic complexity in repeated games. In *Game Theory and Applications* (T. Ichiishi, A. Neyman, and Y. Tauman Eds.), pp. 131–157. San Diego, CA: Academic Press.

Neyman, A. (1997). Cooperation, repetition and automata. In *Cooperation: Game Theoretic Approaches* (S. Hart and A. Mas-Colell Eds.), Volume NATO ASI–Series F, Vol. 155, pp. 233–255. New York, NY: Springer Verlag.

Neyman, A. and D. Okada (1999). Strategic entropy and complexity in repeated games. *Games Econ. Behavior 29*, 191–223, doi:10.1006/game.1998.0674.

Neyman, A. and D. Okada (2000a). Repeated games with bounded entropy. *Games Econ. Behavior 30*, 228–247, doi:10.1006/game.1999.0725.

Neyman, A. and D. Okada (2000b). Two person repeated games with finite automata. *Int. J. Game Theory 29*(3), 309–325.

O'Connell, T. C. (2000). *Bounded Rationality in Repeated Games and Mechanism Design for Agents in Computational Settings*. Ph. D. thesis, Department of Computer Science, University at Albany, SUNY, Albany, NY 12222.

Papadimitriou, C. (1992). On games with a bounded number of states. *Games and Econ. Behavior 4*, 122–131.

Shapley, L. S. and R. N. Snow (1950). "Basic Solutions of Discrete Games". In *Contributions to the Theory of Games, Annals of Mathematical Studies 24* (H. W. Kuhn and A. W. Tucker Eds.), pp. 27–35. Princeton, NJ: Princeton University Press.

Stearns, R. E. (1989). Memory bounded game playing automata. Technical Report No. 547, Institute for Mathematical Studies in the Social Sciences, Stanford University.