On Optimal Game Tree Propagation for Imperfect Players

Eric B. Baum
NEC Research Institute
4 Independence Way
Princeton NJ 0854
eric@research.nj.nec.com

Abstract

We consider the approach to game playing where one looks ahead in a game tree, evaluates heuristically the probability of winning at the leaves, and then propagates this evaluation up the tree. We show that minimax does not make optimal use of information contained in the leaf evaluations, and in fact misvalues the position associated with all nodes. This occurs because when actually playing a position down the game tree, a player would be able to search beyond the boundaries of the original search, and so has access to additional information. The remark that such extra information will exist, allows better use of the information contained in the leaf evaluations even though we do not have access to the extra information itself. Our analysis implies that, while minimax is approximately correct near the top of the game tree, near the bottom a formula closer to the probability product formula is better. We propose a simple model of how deep search yields extra information about the chances of winning in a position. Within the context of this model, we write down the formula for propagating information up the tree which is correct at all levels. We generalize our results to the case when the outcomes at the leaves are correlated and also to games like chess where there are three possible outcomes: Win, Lose, and Draw. Experiments demonstrate our formula's superiority to minimax and probability product in the game of Kalah.

§1: Introduction

It is well known that minimax is an optimal game playing strategy provided the whole game tree can be searched (Von Neumann & Morgenstern 1947). For complex games such as chess, limitations in computational resources allow only a partial search. (Shannon 1950) proposed that a computer might play chess by searching to a depth d, evaluating the positions found using some heuristic evaluation function, and then using minimax to propagate values up the search tree. (Pearl 1984) proposed the probability product rule as an alternative to minimax. This proposal was based on the assumption that the heuristic evaluation function estimates the probability that the position is a win against best play, and then follows from the remark that the correct way to combine (independent) probabilities is by probability product. This reasoning correctly asserts that a position with 100 alternative moves, each of which has independently .1 probability of leading to a won game, is almost certainly a won position. We observe here that this offers little solace to the player in this position, if he can not figure out which moves lead to wins, for he must choose one particular move, and then loses with probabity .9. This paper studies how optimal game play is affected by the computational limitations of the players. Due to these limitations, the game is effectively one of imperfect information, and this information is effectively asymmetric. Neither Minimax nor Probability Product takes account of this. We describe a strategy which does.

Imagine playing the following game, game 1. Player A makes either move $A1$ or move $A2$. Then player B makes either move $B1$ or $B2$. Then player A pays player B \$1 with probability P_{ij} for $i = A1$ or $A2$ and $j = B1$ or $B2$. Let $P_{11} = .5$, $P_{12} = .5$, $P_{21} = .6$, $P_{22} = .1$. Now player A should follow the optimal minimax strategy and make move 1, and the expected payoff to player B will be .5. Now imagine a slightly different game. The rules of game 2 are the same as game 1 except that after player A moves, but before player B moves, the payoffs are generated according to P_{ij}, and player B is told what the outcome of his moves would be. Now if player A chooses move 1, there will be probability $1 - (1 - .5)^2 = .75$ that player B will have a move which wins for him. If player A chooses move 2, however, then the expected payoff for player B is only $1 - .4 \times .9 = .64$. Accordingly player A must change his strategy from the first game.

Now consider playing a game like chess by constructing the game tree to a depth of (say) 10 ply, evaluating the leaves using a necessarily noisy evaluation function, and then propagating the value up the tree by

minimax. We claim this minimax propagation makes suboptimal use of the information contained in the leaf evaluations. The notion of minimax is to assign values to the positions associated with the nodes 9 ply deep by taking the maximum value (to the opponent, player B) of the accessible positions 10 ply deep, just as we assigned a value to player B's position at a depth of 1 ply in game 1 above. But, just as in game 2 above, our opponent has much better information about whether the outcome is a win or a loss for him than is contained in our evaluation function. This is because, when he comes to play in this position, he will search it 10 ply deep! Thus we should assign a value to our opponent, of being in this position, according to a formula closer to the probability formula useful in game 2, than the minimax formula useful in game 1. In general we expect a smooth interpolation between using a formula near probability product deep in the search tree and using a formula near minimax near the top of the tree.

We assume the evaluation function estimates the probability of winning at a position in actual play. In minimax only rank ordering of positions matters, so standard evaluation functions may require monotonic but nonlinear rescaling. One may train evaluation functions to represent probability by accumulating statistics (Pearl 1984).

Besides (Pearl 1984) there have been other previous studies of alternative propagation schemes. (Chi & Nau 1989) claimed, for the game of three hole kalah, that probability product outperforms minimax for the same depth of search. In our experiments, see §5, we found minimax far superior to probability product and attribute their result to the fact that they only worked at depth 2 ply. Another approach was that of (Slagle and Dixon 1969) who remarked that the evaluation function was a noisy version of the truth and that minimax is suboptimal because it ignores the fact that "it is valuable to have several good alternatives in a planned course of action". Their $M\&N$ procedure assigns to a max(min) node some heuristic function of the $M(N)$ highest(lowest) valued successors.

We will not have space here to discuss issues of time complexity beyond remarks in this paragraph. The alpha-beta pruning algorithm computes the exact result of minimax propagation in time roughly the square root of that required by brute force calculation. Now we argue that 'the exact result of minimax propagation' is only an approximation to the correct propagation. Nonetheless, approximate use of information from a larger tree might likely be better than optimal use of information from a smaller tree. For this reason one should not expect practical improvements from the propagation algorithms discussed in this paper without addressing the question of pruning. We are here concerned only in principle with optimal use of the information. Once we understand what calculation is in principle optimal, we may consider how to efficiently approximate it. An expanded version of this paper (Baum 1991) shows that much of the computational benefit of alpha-beta pruning can be taken over into propagation using truncated versions of our formula. Thus one can efficiently compute a better approximation than minimax to the information-optimal propagation. A still more promising approach is also under investigation. Recently a number of authors (McAllester 1988), (Rivest 1988), (Russell & Wefald 1991), have proposed algorithms for growing a search tree including only relevant nodes. Ideas of these authors on pruning can be readily complemented by the results in the present paper on propagation. By combining ideas from these authors, some new algorithmic ideas, and the information theoretic results of the present paper, we believe that improved search algorithms can be produced. This will be discussed elsewhere (Baum & Smith 1992).

§2 derives a formula useful for propagating evaluations up game trees. §3 describes the approximations inherent in the formula of §2 and when it is optimal. §4 discusses generalization to the case of correlated probabilities. §5 describes experiments on the game of Kalah. §6 is a summary.

§2: An Interpolation Between Minimax and Probabilistic Combination

Consider the following one person game, game 3. Player A has b possible moves. A referee declares move $i \in \{1, ..., b\}$ a Win with probability p_i. He does not tell player A these outcomes, however, until after A moves. Instead he supplies to player A b bits of information h_i, where h_i is 1 with probability p^i_W given that move i is a Win, and is 1 with probability $1 - p^i_L$ given that move i is a Loss. Player A knows the p_i and p^i_W and p^i_L.

Game 3 is a model of deep search in a game like chess. Think of h_i as a hint about the value of move i. For appropriate p^i_W and p^i_L, h_i models the extra information in deep search relative to simple evaluation. If $p^i_W = p^i_L = 1$, h_i tells the exact outcome, and game 3 is like game 2. If $p^i_W = p^i_L = \frac{1}{2}$, there is no extra information, and game 3 is like game 1.

What is the value V of game 3 to player A? By Bayes law,

$$P(i = W|h_i = a) = \frac{P(h_i = a|i = W)P(i = W)}{P(h_i = a)}. \tag{2.1}$$

Here $a = 0, 1$. The notation should be evident. For example by $P(i = W|h_i = 1)$ we mean the probability that move i leads to a win given that h_i is 1. Now

$$P(h_i = 1) = p^i_W p_i + (1 - p^i_L)(1 - p_i) \tag{2.2}$$

$$P(h_i = 0) = (1 - p^i_W)(p_i) + p^i_L(1 - p_i) \tag{2.3}$$

so

$$P(i = W|h_i = 1) = \frac{p^i_W p_i}{p^i_W p_i + (1 - p^i_L)(1 - p_i)} \tag{2.4}$$

$$P(i = W|h_i = 0) = \frac{(1 - p_W^i)p_i}{(1 - p_W^i)(p_i) + p_L^i(1 - p_i)} \quad (2.5)$$

We now calculate the probability V of winning for player A who uses Bayes optimal strategy. Let $P(i,a) \equiv P(i = W|h_i = a)$. Order the moves so $P(i,1) \geq P(i+1,1)$ for all i. Let c be the smallest number for which an i satisfies $P(i,0) \geq P(c+1,1)$ (if there is no such number let $c = b$). Let i_0 be arg max $P(i,0)$. Then

$$V = \sum_{j=1,c} p_W^j p_j \prod_{i=1,j-1} (p_i(1 - p_W^i) + (1 - p_i)(p_L^i))$$

$$+ p_{i_0} \frac{(1 - p_W^{i_0})}{p_{i_0}(1 - p_W^{i_0}) + (1 - p_{i_0})(p_L^{i_0})}$$

$$\times \prod_{i=1,c} (p_i(1 - p_W^i) + (1 - p_i)(p_L^i)) \quad (2.6)$$

When $h_1 = 1$, the Bayes optimal choice is move 1 (i.e. eqn's 2.4 and 2.5 imply move 1 is most likely to win, independent of h_i for $i \neq 1$). The payoff then is p_1. This is the first term in the sum. When $h_1 = 0, h_2 = 1$ player A picks move 2. This has payoff p_2 and occurs with probability $(p_1(1 - p_W) + (1 - p_1)(p_L))p_W$ since $h_2 = 1$ with probability p_W when move 2 is a win and $h_1 = 0$ with probability $(1 - p_W)$ when the first move is a Win or with probability p_L when the first move is a Loss. This yields the second term in the sum. Finally if $h_i = 0$ for $i = 1, 2, ..., c$, the Bayes optimal choice is move i_0. This event yields the last term.

Note that we recover the appropriate values in the limiting cases. When player A has no additional information beyond the p_i, i.e. when $\forall i : p_W^i = p_L^i = \frac{1}{2}$, then $c = 1$ and $V = p_1 p_W + p_1(1 - p_W) = p_1$, the minimax value. When $\forall i : p_W^i = p_L^i = 1$, the limit of perfect information, then $c = b$ and we recover the probability formula $V = 1 - \prod_{i=1,b}(1 - p_i)$. Note also that formula 2.6 can easily be computed in time $O(c)$ (once we sort the p_i).

In human chess play, consideration of drawing prospects is important, particularly in endgames. Many standard chess programs seem relatively weak in both their consideration of drawing prospects and their handling of endgames. The arguments of this section can be readily generalized to games with three outcomes: Win, Lose, or Draw. We compute at each position both the probability of Winning and of Drawing and can then make move decisions based on overall Utility of the prospects. Equations analagous to 2.6 can be derived for propagating V_W (the probability of winning) and V_D (the probability of drawing) up the search tree. One may also perform calculations involving conditional probabilities. (Humans will frequently enter into a line of play which they are confident can be drawn but preserves winning chances.) Details are contained in the long version (Baum 1991).

§3: A Model of Deep Search

In this section we will first derive an exact formula, which optimally propagates leaf evaluations up a tree given essentially as much knowledge as one might in principle gather about how time for analysis improves evaluation of outcome. Formula 2.6 will arise as an approximation, making more realistic assumptions about how much we know about how deep search and analysis improves on simple heuristic evaluation. Formula 2.6 will be seen to be exact in a simple model of evaluations studied by (Schrüfer 1986), and for this model we will give the appropriate p_W and p_L.

We wish first to calculate the probability V that our opponent will win in a position where he has b possible moves and we have evaluated by using our simple heuristic evaluation function the probability that he wins if he makes move i as p_i. For generality, let λ_i denote the value(s) of any (collection) of other functions of the position which might be useful. So, e.g., λ might include an estimate of how "tactical" the position is, if we believe deep search to be more informative relative to simple evaluation in "tactical" positions than in more "positional" situations. We will thus evaluate V in terms of (1) our heuristic evaluation of the probability of winning of its successors, and (2) an estimator, which we may previously train on some large set of games, of how much deep search improves evaluation relative to our simple heuristic evaluator, where we allow this estimator to depend on a small number of features. This estimator will be called $P(q|p_i, \lambda_i)$ which we define to be the probability that extended analysis when actually playing this position, say by depth d search, will lead to the estimate that the probability of winning by playing move i is between q and $q + dq$. Now we have

$$V = \int_0^1 dq\, q \times (\text{Probability highest evaluated move has probability } q). \quad (3.1)$$

The highest evaluated move has probability estimate q provided (a) none of the b alternatives has estimate higher than q, and (b) at least one has probability estimate q. (Note this means between q and $q + dq$. For pedagogical clarity we omit epsilantics wherever possible.) The probability that none of the b moves is estimated to have probability higher than q is $[\prod_{i=1,b}(1 - \int_q^1 dq'\ P(q'|p_i, \lambda_i))]$. The probability that move i has evaluation q, given that it does not have evaluation higher than q is $\frac{P(q|p_i,\lambda_i)}{(1-\int_q^1 dq'\ P(q'|p_i,\lambda_i))}$. Thus the probability that at least one of the b moves has probability q, given that none of them has evaluation higher than q, is $\{1 - \prod_{i=1,b}(1 - \frac{P(q|p_i,\lambda_i)}{(1-\int_q^1 dq'\ P(q'|p_i,\lambda_i))})\}$. Putting this together we find

$$V = \int_0^1 dq\ q[\prod_{i=1,b}(1 - \int_q^1 dq'\ P(q'|p_i, \lambda_i))]$$

$$\times\{1-\prod_{i=1,b}(1-\frac{P(q|p_i,\lambda_i)}{(1-\int_q^1 dq'\ P(q'|p_i,\lambda_i))})\} \quad (3.2)$$

More generally, we must compute the probability of winning at a node at level $l+1$ in the search tree, given that we have calculated, by propagation up from the leaves, that the probability of winning if we choose move i is p_i^l. To do this we measure the distribution $P(q|p_i^l,\lambda_i)$, and we have simply

$$V=\int_0^1 dq\ q[\prod_{i=1,b}(1-\int_q^1 dq'\ P(q'|p_i^l,\lambda_i))]$$

$$\times\{1-\prod_{i=1,b}(1-\frac{P(q|p_i^l,\lambda_i)}{(1-\int_q^1 dq'\ P(q'|p_i^l,\lambda_i))})\} \quad (3.3)$$

Unfortunately the utility of this formula is limited by any practical limitations in our ability to approximate $P(q|p_i^l,\lambda_i)$. We propose equation 2.6 as a simple approximation which is easy to use and reasonably accurate. The approximation inherent in 2.6 is that q can take on only one of two values, given p_i and λ_i. Thus $P(q|p_i^l,\lambda_i)$ is simply the sum of two delta functions. These two values are given by eqns. 2.4 and 2.5. One value (corresponding to $h_i=1$) is a revised estimate from depth d search that the probability of winning is higher than depth l search estimates, and the other is a revised estimate that the probability is lower.

In fact equation 2.6 can be seen to be exact in the widely studied model where the evaluation function is Boolean, returning either 0 or 1, and where our knowledge of correctness of our evaluation is dependent only on depth of search, and not on other features of the position. This second condition occurs, e.g., if we neglect (as is common in game programs) to estimate feature dependence of the accuracy of our calculation. Under these assumptions, if depth d search, returns a 1 (respectively a 0), we assign a certain probability p_d^+ (respectively p_d^-) that the position is Won. Accordingly $P(q|p_i^l)$ is bivalued and equation 2.6 exact. These conditions hold, e.g. in the model studied by (Schrüfer 1986) and in the related model of (Nau 1983).

(Schrüfer 1986) studied the following model. We build a game tree starting from the root, which may be either a Win or a Loss node, and grow a tree with uniform branching ratio b. Below every Win node there are n Loss nodes and $b-n$ Win nodes (where we define node values with respect to the player on move). By definition, a Win node must have at least one winning option, so $n\geq 1$. Below every Loss node, again by definition, all b children are Wins. The tree is built to depth d. A heuristic evaluation function returns a noisy estimate of the true value of the leaves as follows. If the leaf is a Win node, the evaluation is $e=1$ with probability $1-e_0^+$ (and thus 0 with probability e_0^+) and if the leaf is a Loss node, the evaluation is 0 with probability $1-e_0^-$ (and thus 1 with probability e_0^-).

Now (Schrüfer 1986) derives recursion relations relating the probability of error at level $l+1$ above the leaves to that at level l (denoted $e_l^\pm$), assuming the evaluation is propagated up the tree by minimax:

$$e_{l+1}^+=(e_l^-)^n(1-e_l^+)^{b-n}\ ;\ e_{l+1}^-=1-(1-e_l^+)^b \quad (3.4)$$

Analyzing these equations, (Schrüfer 1986) showed that deep search is pathological (i.e. gets worse the deeper you search) provided $n=1$, but is at least exponentially convergent to zero error (i.e. $e^+\sim 0$ and $e^-\sim 0$) if $n>1$ and e_0^+ and e_0^- are sufficiently small. (Schrüfer 1986) also analyzed a somewhat more complex model, where n is an i.i.d. random variable at each Win node in the tree, chosen with probability p_j to be j for $j=1$ up to b. These results cast considerable light on the question, hotly debated in the theoretical AI literature for ten years, of when game search is pathological.

In Schrüfer's model we may calculate how deep search improves the accuracy of our evaluation. Also, since the evaluation function is bivalued, and since accuracy depends only on depth of search (rather than say other parameters of the position or game tree) the evaluator $P(q|e_l)$ is is in fact zero except for two values of q. In (Baum 1991) we calculate explicitly in this model $P(q|e_l)$ and show that equation 3.3 reduces exactly to equation 2.6 for appropriate values of p_W and p_L. One demands that p_W and p_L be chosen so that the probability of winning given by eqns 2.4-5 is equal to that given by depth d search. We omit the lengthy Bayesian calculation for reasons of space but state the values of p_W and p_L which emerge. We find

$$p_W^1(l)=\frac{(1-e_d^+)(1-e_l^+-e_d^-+e_l^+e_d^--e_l^-e_d^+)}{(1-e_l^+)(1-e_d^+-e_d^-)} \quad (3.5)$$

$$p_L^1(l)=\frac{(1-e_d^-)(e_l^--e_d^-+e_l^+e_d^--e_l^-e_d^+)}{(e_l^-)(1-e_d^+-e_d^-)} \quad (3.6)$$

$$p_W^0(l)=\frac{(1-e_d^+)(e_l^+-e_d^++e_l^-e_d^+-e_l^+e_d^-)}{(e_l^+)(1-e_d^+-e_d^-)} \quad (3.7)$$

$$p_L^0(l)=\frac{(1-e_d^-)(1-e_l^--e_d^++e_l^-e_d^+-e_l^+e_d^-)}{(1-e_l^-)(1-e_d^+-e_d^-)} \quad (3.8)$$

These formulae tell us how to calculate the p_W and p_L and use formula 2.6. Here the superscripts on p_W and p_L correspond to the superscripts i in eqn 2.6 as follows. One should use superscript 1 (eqns 3.5-6) in equation 2.6 for moves i with $p_i\geq\frac{1}{2}$ and one should use superscript 0 (eqns 3.7-8) for moves with $p_i<\frac{1}{2}$. Recall that e_l^+ (resp. e_l^-) is defined as the probability that a depth l search predicts a state is lost (resp. won) when in fact it is won (resp. lost) (and $e_d^\pm$ is simply $e_l^\pm$ for $l=d$). The $e_d^\pm$ and $e_l^\pm$ are thus directly measurable by taking statistics. Alternatively, they may be calculated in Schrüfer's model. If our search is accurate, it will be a good approximation to neglect both $e_d^\pm$ and $e_l^\pm$ with respect to 1. This yields:

$$p_W^1=1,\ p_L^1=1-\frac{e_d^-}{e_l^-};\ p_W^0=1-\frac{e_d^+}{e_l^+},\ p_L^0=1 \quad (3.9)$$

Thus convergence to perfect information is linear in the ratio of the accuracy of depth d search to depth l search. (Schrüfer 1986) shows that, when deep search is not pathological, it is typically at least exponentially convergent. Thus p_W and p_L are likely near one for many levels in a search tree.

§4: Correlations

So far we have proceeded under the assumption that that probabilities p_i of winning are independent. Since these are probabilities of winning of positions differing by only two moves, they more realistically should be considered correlated. Note that we are not concerned with correlations in the distribution of the values of the p_i. This kind of correlation implies that if $p = .9$ for a position, than a sibling position is likely to have $p \geq .5$. This must occur, for if nearby positions are not more likely to lead to similar results, we could not play using a heuristic evaluation function. Instead we are discussing now correlations between the outcomes, given the probabilities. This kind of correlation occurs when two siblings each have $p = .8$, but the probability that *either* sibling wins is less (or more) than $1 - .2^2$.

In (Baum 1991) we give a more general discussion of correllations. Here for conciseness we specialize to a simple assumption about the form of correlations. This assumption seems plausible as an approximation, and is simple enough to allow a readily computable formula. Since we have little idea about the nature of the correlations in practice it also seems reasonable to choose a model which allows a one parameter fit. Our simplified model follows. Correlation is specified by one parameter: γ. Given $p_1 \geq p_2 \geq ... \geq p_k$ we determine which moves lead to a win as follows. A referee first draws a bit B_C which has probability γp_k of being 1. If B_C is a 1, then all moves, 1,...,k are W. If B_C is a 0, then move i : $i = 1, ..., k$ is a W with probability $p_i - \gamma p_k$, where all of these are now determined independently. Note we don't care what happens for $i > k$, because we assume either that these moves are totally correlated with move 1, or because in any case the combination of their correlation plus low probability means the player should never select them, independent of the h_i, or because we simply consider strategies which truncate after considering the k best moves. If we only consider strategies which truncate after 2 moves (i.e. if k=2) this model of correlations can be easily seen to be without loss of generality.

Now it is easy to write down the analogue of equation 2.6. Let $V(p_1, p_2, ..., p_k)$ be defined by equation 2.6, where $c \rightarrow k$. For $i = 1, ..., k$ let

$$p_i' \equiv \frac{p_i - \gamma p_k}{1 - \gamma p_k}. \tag{5.1}$$

Call the value of the position, in the model with correlations, V'. Then

$$V' = \gamma p_k + (1 - \gamma p_k) V(\vec{p}'). \tag{5.2}$$

This is evident since with probability γp_k, $B_C = 1$, and the position is a win. Otherwise, the probability of winning is $V(\vec{p}')$. Note that in the limit of perfect correlation, i.e. $\gamma = 1$ and $p_k = p_2$, we recover the minimax value p_1, since then $p_2' = 0$, and so we discover in computing $V(\vec{p}')$ that $c = 1$ and 5.2 reduces to $\gamma p_2 + (1 - \gamma p_2) p_1' = p_1$.

§5: Experiments with Kalah

This section describes preliminary experiments[1] comparing performance of programs using minimax, probability product, and formula 2.6 on the game of Kalah. Kalah has been described as a "moderately complex game, perhaps on a par with checkers" (Slagle & Dixon 1970). See (Slagle & Dixon 1969) for a discussion of the rules. We mention here only that the object of Kalah is to capture stones.

As evaluation function (Slagle & Dixon 1969) proposed and (Chi & Nau 1989) studied Kalah Advantage (KA) defined as the difference in number of stones currently captured (scaled to be between 0 and 1 to estimate a probability of winning). We found this did not adequately estimate the probability of winning. Instead we used the following. Define S≡KA/NR where NR is the number of stones remaining on the board (and thus potentially winnable). Our evaluation function then was $E \equiv (1 - S)/2$. In head to head competition using depth 6 ply alpha-beta, E played a substantially stronger game than KA. Note that when KA=NR (so that a mathematical win or loss has occurred) E is 1 or 0, and when KA=0, $E = 1/2$. This is a reasonable estimate of the probability of winning for KA=0, but not precise since the side on move has an advantage (about 53% in our tournament). More comprehensive experiments might take statistics to find a nonlinear, monotonic rescaling of E which more closely estimated probability of winning. This would presumably increase the edge of formula 2.6.

We simply chose to set, for ϵ a parameter:

$$p_W = p_L = 1 - \frac{1}{2}(\epsilon^{d-l})$$

where $d = 6$ was the depth of our search and $l = 0, ..., 4$ was the level above the leaf. The exponential form was suggested by the dependence of p_W and p_L in equations 3.9 on e_l and e_d and the exponential increase in accuracy of deep search in Schrüfer's model. Note that for this first cut we approximated by setting $p_W = p_L$ and by setting both independent of all other factors, including p_i, the probability the move will win. More accurate tuning up of the p_W^i and p_L^i would be possible with more work. Presumably this would improve the performance of formula 2.6. A small tournament was played on 400 games for values of $\epsilon = .8, .85, .9$, and .95. Formula 2.6 seemed to have an edge against minimax

[1]These experiments were performed by E. Wigderson using code partially written by W. D. Smith.

for each of these values, but the edge was largest for $\epsilon = .9$ so we then performed a tournament of 20,000 games for this value. Note $\epsilon = .9$ corresponds to relatively modest choices of p_W and p_L. By parametrizing p_W and p_L in this way and choosing ϵ to maximize results, we implicitly tuned against the degree of correlation, and the modest values of p_W and p_L may reflect a fair degree of correlation.

Tournaments were performed by choosing 10,000 starting positions by making the first 6 moves at random. For each of these starting positions competitor A and B played both First and Second, where A and B could be any of minimax, formula 2.6, or probability product(PP). All algorithms played to depth 6 ply (full width). Formula 2.6 won 9906 games against minimax, while minimax won 9171, and 923 were draws. Minimax won 18359 games against PP, while PP won 1581, and 60 were drawn. 2.6 won 18882 against PP, while PP won 1040, and 78 were drawn. Note that probability product was awful, which is attributable to its poor treatment of correlations (Smith 1992). Note that (excluding draws) formula 2.6 beats minimax $52 \pm .03\%$ of the games. This is a small but clear edge (significant at more than 5σ) which possibly could be improved by better tuning of the evaluation function to reflect probability, better tuning of p_W^i and p_L^i, and using a formula such as equation 5.2 which explicitly accounts for correlations. Note as discussed in §1, that this paper does not reflect time performance issues, but merely performance on equal depth trees.

§6: Discussion

We have remarked that minimax incorrectly values positions since it does not account for extra information that players will have when they actually encounter the position. This extra information arises, for example, from their ability to do a deep search from the position. While it is not possible to know the "true" value of a position without actual possession of this extra information (and perhaps some in addition), the mere realization that the players will act based on extra information allows more accurate estimation of the value of positions. A similar phenomenon will arise in search and planning contexts other than games. When computational or informational limitations force one to make a heuristic choice, one's choice should take account of the relative computational advantages in future choices.

We have given a formula (3.3) which correctly propagates up the tree the information contained in evaluations of the probability of winning at the leaves given detailed, but in principle available, information about the nature of extra information (under the assumption of independence of the leaf probabilities). Since this detailed information seems likely to be difficult to obtain in practice, we have proposed a simple formula for propagating information up the game tree. This model requires only limited, and readily obtainable estimates of the extra information. It is exact in widely studied models (e.g. Schrüfer 1986.) Experiments in the game of Kalah exhibit the superiority of this formula to both minimax and probability product.

Acknowledgement: I thank L.E. Baum for suggesting that I consider games with draws, W. D. Smith for helpful comments on a draft, and especially E. Wigderson and W. D. Smith for the experimental work reported in §5.

References

Baum, E.B. 1991. "Minimax is not optimal for imperfect game players", preprint, submitted for publication.

Baum, E.B., and W. D. Smith. In preparation.

Chi, P-C, D. S. Nau. 1989. "Comparison of the Minimax and Product Back-up Rules in a Variety of Games", in *Search in Artificial Intelligence*, eds. L. Kanal and V. Kumar, Springer Verlag, New York, pp451-471.

McAllester, D. A. 1988. "Conspiracy numbers for minmax search", Artificial Intelligence v35 pp287-310.

Nau, D. S. 1983. "Decision Quality as a Function of Search Depth on Game Trees", Journal of the ACM, V 30, No. 4, pp 687-709.

Pearl, J. 1984. *Heuristics, Intelligent Search Strategies for Computer Problem Solving*, Addison Wesley Publishing Co, Reading MA.

Rivest, R. L. 1988. "Game Tree Searching by Min/Max Approximation", Artificial Intelligence 34 pp77-96.

Russell, S., and E. Wefald. 1991. *Do the Right Thing, Studies in Limited Rationality*, MIT Press, Cambridge MA.

Schrüfer, G. 1986. "Presence and Absence of Pathology on Game Trees", in D.F. Beal, ed., *Advances in Computer Chess* 4, (Pergamon, Oxford, 1986) pp 101-112.

Shannon, C. E. 1950. "Programming a Computer for Playing Chess", Philosophical Magazine 41(7): 256-75.

Slagle, J. R., and J. K. Dixon. 1969. "Experiments with some programs that search game trees", JACM V16, No 2 pp 189-207.

Slagle, J. R., and J. K. Dixon. 1970. "Experiements with the *M&N* Tree-Searchin Program, CACM 13(3)147-154.

Smith, W.D., 1992. personal communication.

Von Neumann, J., and O. Morgenstern. 1947. *Theory of Games and Economic Behavior* Princeton University Press, Princeton.