Conspiracy Numbers for Min-Max Search*

David Allen McAllester
Department of Computer Science, Cornell University, Ithaca, NY 14853, U.S.A.

Recommended by Judea Pearl

ABSTRACT

A new procedure is presented for growing min-max game trees. In large games, such as chess, decisions must be based on incomplete search trees. The new tree-growth procedure is based on "conspiracy numbers" as a measure of the accuracy of the root minimax value of an incomplete tree. Conspiracy numbers measure the number of leaf nodes whose value must change in order to change the minimax root value by a given amount. Trees are grown in a way that maximizes the conspiracy required to change the root value. The trees grown by this procedure are often deep and narrow. However, if all static values in a game are the same, this new procedure reduces to d-ply search with α-β pruning. Unlike B* *search, nonuniform growth is achieved without any modification of the static-board evaluator.*

1. Introduction

The game playing procedure presented here uses minimax search: the procedure grows a search tree and computes minimax values for all non-leaf nodes in the tree. A move is then chosen based on these minimax values. However, a novel technique is used for growing the search tree and the resulting trees are shaped in complex and nonuniform ways. The new tree-growth procedure is based on a measure of the "accuracy" of the minimax root values of incomplete search trees. Search trees should be grown so that the minimax root values are accurate.

Consider the family of full d-ply search trees. A full d-ply search tree is one which contains the set of all positions which are within d ply of a given root position. It is generally agreed that for the game of chess a $(d+1)$-ply tree is

* This paper describes work done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-86-K-0180.

Artificial Intelligence **35** (1988) 287–310
0004-3702/88/$3.50

more likely to have an accurate minimax root value that a d-ply tree; computer programs which search $(d+1)$-ply trees play better than programs which search d-ply trees. However, there is currently no satisfactory theoretical analysis of this phenomenon. In fact Nau [1, 2] has shown that for certain pathological games $(d+1)$-ply search trees are *less accurate* than d-ply search trees. It is also well known that "flat" d-ply trees do not provide the best performance in chess playing programs. All effective modern programs explore capture moves to a greater depth than they explore non-capture moves and moves resulting in check are not counted as a ply of search (see Berliner [3]). Thus it is clear that the accuracy of the minimax root value of a search tree can depend on both the size and the shape of the tree. But is there some general way of measuring the accuracy of the mimimax root value of incomplete trees?

Intuitively a search tree has an accurate minimax root value if that value would not change much if the tree were expanded further; a root value is unreliable if further search results in a major change in that value. Thus the term "stable" in perhaps better than the term "accurate". One should grow a search tree in such a way that the root value is stable.

The stability of the root value can be measured in terms of *conspiracy numbers*. A conspiracy number measures the number of leaf nodes whose value must be changed in order to change the root node's value by a certain amount. A change in the value of a certain set of leaf nodes is called a conspiracy between those leaf nodes. A conspiracy number measures the size of the conspiracy needed to bring about a certain change in root value; the more conspirators needed for a given change the less likely the change.

In a game of chess each player has many choices. If a certain move turns out to be bad for white, then white can usually make some other move and avoid the bad position. This ability to avoid bad positions means that many different positions must turn out to be bad before a given player can be forced into a bad position. Now consider the conspiracy numbers mentioned above. Since each player has many choices many different leaf positions must all conspire to be bad before the minimax value of the root position becomes bad. Thus a significant change in the root value often requires a conspiracy between a large number of leaf nodes. On the other hand there are certain positions where a given player has very little choice. Suppose for example that the person playing white can "force" the person playing black to make a certain series of moves. If the position which is reached as a result of these moves turns out to be bad for black, then the original position will turn out to be good for white. In situations where the black player does not have much choice only a small conspiracy between leaf nodes is needed to make the root position good for the white player. The tree-growth procedure presented here explores such "forced" lines more deeply than lines in which both players have a lot of choice.

Given a particular search tree the conspiracy numbers computed from that tree can be used to compute a range of likely root values. More specifically

consider a potential root value V. We say that V is a likely root value if the conspiracy required to convert the actual root value to V is below a certain threshold. As the potential values get farther from the current minimax root value the conspiracies required to achieve those values get larger. Thus the set of likely root values for a tree T is an interval of the form $[V_{min}, V_{max}]$. The "accuracy" of the current minimax root value can be judged by computing the range $[V_{min}, V_{max}]$ of likely root values.

The range of likely root values $[V_{min}, V_{max}]$ computed from conspiracy numbers is quite different from the range $[\alpha, \beta]$ used in α-β pruning (see Knuth [4]) and the SSS* algorithm (see Stockman [5]). The parameters α and β provide exact information about the complete d-ply search tree under a given node. Using the parameters α and β one can compute the exact minimax root value of the complete d-ply tree without actually examining all of the tree. On the other hand the range $[V_{min}, V_{max}]$ computed from conspiracy numbers does not provide exact information about any particular finite search. Instead the range $[V_{min}, V_{max}]$ provides a heuristic measure of the accuracy of the minimax root value of any incomplete search tree. Suppose that the entire d-ply search tree has been explored. One can compute a range of likely root values for this full tree and this range may contain more than one value. The full d-ply search tree is not guaranteed to be accurate.

Trees should be grown in a way that quickly reduces the range of likely root values. The tree-growth procedure presented here takes an arbitrary incomplete search tree and decides which leaf node should be expanded next. Suppose that the range of likely root values for the tree under consideration is $[V_{min}, V_{max}]$. To choose a leaf node for expansion one first chooses either V_{min} or V_{max} as a candidate value to be ruled out. To rule out the value V_{max} one must ensure that there are strategies which the minimizing player can use to avoid V_{max}. Similarly, to rule out V_{min} one can expand leaf nodes which are involved in strategies that the maximizing player can use to avoid the value V_{min}.

Berliner [7] has proposed a search technique called B* which is also based on iterative incremental tree growth. The B* method for tree growth, however, is quite different from the method proposed here. Both procedures manipulate ranges of possible values at nodes of the tree. However, B* and conspiracy number techniques use different methods for computing the value ranges and different methods for selecting the next node to explore. Furthermore, the ranges used in B* are proved correct: the true value is always within the given range. The ranges based on conspiracy numbers are heuristic: a node may sometimes have a true value outside of the computed range. Finally, conspiracy numbers have an advantage over B* in that conspiracy numbers can be used without any modification of the static position evaluator.

Under certain conditions conspiracy-based search bears a strong similarity to d-ply search with α-β pruning. In particular, if all the positions have exactly

the same static value, then the conspiracy-based search is essentially the same as d-ply search with α-β pruning. However, if the static values are not uniform, then the tree grown using the new procedure will be nonuniform: different portions of the tree will be explored to different depths.

For a given bound on computational resources, such as the number of nodes which can be explored or the time allotted the search process, the new procedure is capable of growing trees which are much deeper than the trees which are grown using d-ply search with α-β pruning. Intuitively the procedure gradually rules out "bad" moves near the root of the tree and thus reduces the branching factor at the root position. Unfortunately however there is no *proof* that the conspiracy numbers used in the new procedure provide a good basis for selectively growing search trees. Some experiments with random trees, however, indicate that conspiracy-based search is superior to iteratively deepened α-β search. Ultimately the performance of this procedure must be determined by constructing game playing programs.

2. Conspiracy Numbers

We are interested in two-person perfect information games. It is assumed that each position in such a game is associated with a static value. The two players, max and min, make alternate moves; max tries to maximize the values while min tries to minimize values.

Definition 2.1. A *min-max game* is a tree G where each node of the tree is associated with a number called the *static value* which is either a real number or one of the special values $-\infty$ or $+\infty$. The root node is defined to be of type max; every successor of a max node is of type min and every successor of a min node is type max. A node of G with no successor nodes will be called a *terminal node* of G.

The game of chess forms a min-max game under the definition given above (assuming that some appropriate static-board evaluator is provided). Note that there is no constraint on static values; the static value of a max node need not be the maximum of the static values of its successors. In practice it is impossible to search the entire game of chess. For this reason it is assumed here that the game under consideration is *infinite*. Since the minimax procedure cannot be directly applied to an infinite game one must consider finite subsets of the game G.

Definition 2.2. Let T be any finite subtree of a min-max game G. A node j of T is called a *leaf node* of T if no successor of j is a member of T. The finite subtree T will be called *well-formed* if there are no "partially expanded" nodes, i.e. for every non-leaf node j in T all successors of j are also in T.

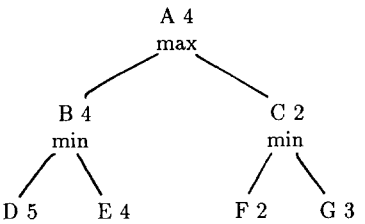

Fig. 1. A simple two-ply tree.

Conspiracy numbers measure the difficulty of changing the minimax root value of a given tree. Consider the simple two-ply tree shown in Fig. 1. The nodes of the tree are given single letter names in an essentially random order (nodes will be named in the order in which they are added to the tree). The minimax value of each node is shown next to the node. The minimax value of the root node A is a function of the values assigned to the leaf nodes D, E, F and G. Changing the value of E to 5 would increase the root value to 5. However changing the value of E to 6 would only increase the root value to 5 because the min player would then choose node D over node E. To change the root value to 6 one would have to change the values of both nodes D and E; changing the root value to 6 requires a "conspiracy" between nodes D and E (or a conspiracy between F and G). The number of conspirators required to change the root value to a given value V provides a measure of the difficulty of changing the root value to V. This measure is not restricted to the root node; for any node j in a tree T one can determine the number of leaf nodes which must conspire to change the value of j to V. These observations lead to the following definition:

Definition 2.3. Let i be any node of the subtree T and let V be any possible node value. A *set of conspirators* for i, T, and V is a set $\mathscr{C}$ of leaf nodes of T which does not contain any terminal nodes of the game G, i.e. every node in $\mathscr{C}$ can be expanded, and by changing the values associated with the nodes in $\mathscr{C}$ it is possible to change the minimax value of the node i to V. A *minimal set of conspirators* for i, T, and V is a set of conspirators $\mathscr{C}$ for i, T, and V such that every other set of conspirators for i, T, and V has at least as many members as $\mathscr{C}$. The *conspiracy number* for i, T, and V, denoted Δ-needed(i, T, V), is the size of a minimal set of conspirators for i, T and V. If there is no set of conspirators for i, T, and V (due to terminal nodes in the game G), then Δ-needed(i, T, V) is defined to be $+\infty$.

The number Δ-needed(i, T, V) measures the difficulty of changing the minimax value of i to the value V; changing the value of i to V requires a conspiracy of at least Δ-needed(i, T, V) leaf nodes. If the minimax value of i is

equal to V, then Δ-needed(i, T, V) is zero. For computational purposes it will be convenient to distinguish between the difficulty of *increasing* the minimax value of a node and the difficulty of *decreasing* the minimax value of a node.

Definition 2.4. The expression $\uparrow$needed(i, T, V) denotes the least number of conspirators needed to *increase* the value of i to V; if the minimax value of i is greater than or equal to V, then $\uparrow$needed(i, T, V) is zero, otherwise $\uparrow$needed(i, T, V) equals Δ-needed(i, T, V).

Similarly, $\downarrow$needed(i, T, V) denotes the number of conspirators needed to *decrease* the value of i to V; if the minimax value of i is already less than or equal to V, then $\downarrow$needed(i, T, V) equals zero, otherwise $\downarrow$needed(i, T, V) equals Δ-needed(i, T, V).

One would expect that as V gets larger it is more difficult to increase the value of a node to V. Similarly, as V gets smaller it should be more difficult to decrease the value of a node to V. This intuition is borne out by the following theorem:

Monotonicity Theorem. *The functions* $\uparrow$needed(i, T, V) *and* $\downarrow$needed(i, T, V) *are monotone in* V; *the function* $\uparrow$needed(i, T, V) *is nondecreasing in* V *and the function* $\downarrow$needed(i, T, V) *is nonincreasing in* V.

The above theorem is implied by the following lemma:

Lemma 2.5. *Let* V_A *denote the minimax root value of the tree* T *and let* V *be any value. If* $\mathscr{C}$ *is a set of conspirators for* i, T *and* V, *then the nodes in* $\mathscr{C}$ *can conspire to produce any root value between* V_A *and* V.

It is difficult to obtain a precise analysis of the significance of the conspiracy number $\uparrow$needed(A, T, V) as a measure of the likelihood that the "true value" of A is greater than or equal to V. No assumption has been made here about the relationship between the static values of neighboring nodes in the game G. If there is really no relationship between the static-board values of neighboring nodes, then no incomplete tree has a stable root value: the result of a d-ply search would provide no information about the result of a $(d+1)$-ply search. In the game of chess however there is clearly a relationship between the static values of nearby nodes.

3. The Range of Likely Root Values

As a search tree is grown its root value can change. A tree T is stable provided that the root value does not change much as the tree is expanded further. Conspiracy numbers can be used to bound the amount of expected variation in

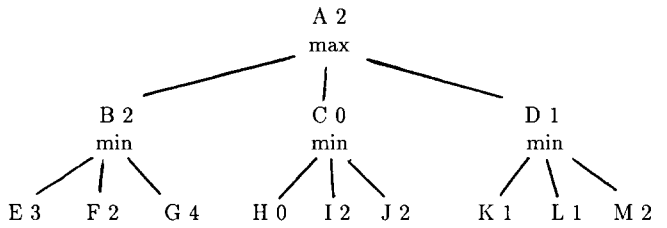

Fig. 2. Values outside the range [1, 3] require two of more conspirators.

the root value. More specifically, for any tree T one can compute bounds $V_{\min}$ and $V_{\max}$ such that the root value is not likely to go below $V_{\min}$ and not likely to go above $V_{\max}$.

Let T be the tree shown in Fig. 2. Increasing the value of node F to 3 will result in increasing the value of the root node A to 3. However increasing the root value above 3 would require a conspiracy between nodes E and F. Increasing the root value above 4 requires a three-way conspiracy, e.g. E, F, and G or H, I, and J. On the other hand, node E can act alone to decrease the root value to 1. Decreasing the root value below 1 requires a two-way conspiracy, e.g. nodes E and K. Decreasing the root value below 0 requires a three-way conspiracy, e.g. nodes E, H, and K. In summary, for the tree shown in Fig. 2, values in the range [1, 3] can be achieved with a single conspirator; values in the range [0, 4] can be achieved with two conspirators; and any value can be achieved with three conspirators.

Definition 3.1. Let i be a node in a partial search tree T. The *upper bound* of node i with respect to conspiracy size N, denoted $V_{\max}(i, T, N)$, is the largest value for node i that can be achieved by N conspirators.

Similarly, $V_{\min}(i, T, N)$ is defined to be the least value for i achievable by N conspirators.

Static values are allowed to be arbitrary real numbers and a set of real numbers need not have a greatest member. However, there always exists a greatest and least value for a node i that is achievable by N conspirators (the proof is omitted here). Furthermore, the greatest and least achievable values for i are either infinite or equal to the static value of some node in the tree rooted at i (again the proof is omitted).

The least and greatest achievable values for a node i with respect to a conspiracy size N determines a range of "likely values" for the node i. If conspiracies of more than N nodes are considered unlikely, then the true value of node i is likely to be in the interval

$$[V_{\min}(i, T, N), V_{\max}(i, T, N)] .$$

4. Computing Bounds

This section presents a technique for directly computing bounds of the form $V_{\max}(i, T, N)$. This section also shows how conspiracy numbers of the form $\uparrow\text{needed}(i, T, V)$ can be derived from the computed bounds. In this section we will only consider upper bounds and conspiracy numbers for achieving values greater than the node's value; lower bounds and conspiracy numbers for achieving values less than the node's value can be computed with the dual algorithm which is obtained by interchanging min and max, reversing the direction of all inequalities between board values, and interchanging the vertical arrows $\uparrow$ and $\downarrow$.

Consider an arbitrary node i in a tree T and consider the sequence of bounds:

$$V_{\max}(i, T, 0)\,,\quad V_{\max}(i, T, 1)\,,\quad V_{\max}(i, T, 2)\,,\quad V_{\max}(i, T, 3), \ldots .$$

This is a nondecreasing sequence whose first value is the minimax value of i. Furthermore, the above sequence eventually takes on the constant value $+\infty$; when N equals the total number of leaf nodes in the tree rooted at i, then $V_{\max}(i, T, N)$ must be $+\infty$. Thus there are only a finite number of finite values in the above sequence and one can explicitly store this list of finite values at every node in the search tree.

Definition 4.1. The *upper bounds sequence* for a node i in a tree T, denoted $\uparrow\text{bounds}(i, T)$, is the initial sequence of finite values of the infinite sequence

$$V_{\max}(i, T, 0)\,,\quad V_{\max}(i, T, 1)\,,\quad V_{\max}(i, T, 2)\,,\quad V_{\max}(i, T, 3), \ldots .$$

Conspiracy numbers of the form $\uparrow\text{needed}(i, T, V)$ can be easily computed from the bounds sequence $\uparrow\text{bounds}(i, T)$. More specifically we have the following lemma:

Bounds Lemma. *For any non-leaf node i in a tree T, the conspiracy number $\uparrow\text{needed}(j, T, V)$ equals the number of elements of the sequence $\uparrow\text{bounds}(j, T)$ which are strictly less than V.*

Proof. Recall that $V_{\max}(j, T, N)$ is the largest value for j that can be achieved by N conspirators. This definition implies the following relations:

$$V_{\max}(j, T, N) < V \quad \text{implies} \quad \uparrow\text{needed}(j, T, V) > N\,,$$

$$V_{\max}(j, T, N) = V \quad \text{implies} \quad \uparrow\text{needed}(j, T, V) = N\,,$$

$$V_{\max}(j, T, N) > V \quad \text{implies} \quad \uparrow\text{needed}(j, T, V) \leq N\,.$$

These relations together imply that $\uparrow\text{needed}(j, T, V)$ is the first integer N such

that $V_{max}(j, T, N) \geq V$. Furthermore, the bounds sequence $\uparrow$bounds(j, T) starts with the value $V_{max}(j, T, 0)$. This implies that, if N is the first integer such that $V_{max}(j, T, N) \geq V$, then N equals the number of elements of the sequence $\uparrow$bounds(j, T) which are strictly less than V. □

The bounds sequence $\uparrow$bounds(i, T) can be easily computed if i is a leaf node. More specifically, if i is a leaf node, then one conspirator can increase i to $+\infty$ so $V_{max}(i, T, 1)$ is $+\infty$. Thus, if i is a leaf node of T, then $\uparrow$bounds(i, T) consists of the single value $V_{max}(i, T, 0)$ which equals the static value of i.

Now suppose i is a non-leaf max node. Recall that $V_{max}(i, T, N)$ is the greatest value that can be achieved by N conspirators. N conspirators can increase the value of i to V just in case N conspirators can increase the value of a single successor node to V. Thus the greatest value for i achievable by N conspirators is the greatest value achievable for any single successor node. This gives the following relation where $S(i)$ is the set of successor nodes of i:

$$V_{max}(i, T, N) = \operatorname*{Max}_{j \in S(i)} V_{max}(j, T, N) .$$

This relation allows the bounds sequence $\uparrow$bounds(i, T) to be computed from the bound sequences of the successors of i. Since the bounds sequence is defined to contain only finite values, the length of the bounds sequence $\uparrow$bounds(i, T) is the minimum length of the bound sequences of the successor nodes of i.

The bounds sequence $\uparrow$bounds(i, T) can also be easily computed in the case where i is a min node. A bounds sequence can be viewed as a multiset: a set in which a given value can appear more than once. Furthermore, any finite multiset of real numbers can be viewed as a bound sequence: any finite multiset of real numbers can be placed in nondecreasing order yielding a bounds sequence. The following theorem provides a way of computing $\uparrow$bounds(i, T) when i is a min node.

Bounds Multiset Theorem. *If i is a non-leaf min node in a tree T, then the bound sequence $\uparrow$bounds(i, T) equals the multiset union of the bound sequences $\uparrow$bounds(j, T) where j is a successor of i.*

Proof. Let i be a non-leaf min node of a tree T. Now, for an arbitrary value V consider the conspiracy number $\uparrow$needed(i, T, V). Since i is a min node, increasing the value of i to V requires increasing the value of every successor of i to V. Thus the number of conspirators needed to achieve V at node i is the sum of the number of conspirators needed to raise every successor of V to i. Thus we have the following relation where $S(i)$ is the set of successors of i:

$$\uparrow\text{needed}(i, T, V) = \sum_{j \in S(i)} \uparrow\text{needed}(j, T, V) .$$

Now, by the Bounds Lemma, for each successor node j we know that $\uparrow$needed(j, T, V) equals the number of elements of $\uparrow$bounds(j, T) which are less than V. Now let B be the multiset union of the bounds sequences of the successor nodes. The above equation for computing $\uparrow$needed(i, t, V) implies that $\uparrow$needed(i, t, V) equals the number of elements of B which are less than V. Now assume that B is placed in nondecreasing order, i.e. B consists of the values $B_0, B_1, B_2, \ldots, B_k$. We must show that B is the bounds sequence for node i.

First suppose $N \leq k$. We wish to show that B_N equals $V_{\max}(i, T, N)$, i.e. that B_N is the largest value achievable by N conspirators. By the above comments a value V is achievable by N conspirators just in case there are N or fewer values of B less than V. It is easy to show that B_N is the largest value with this property.

Finally we must show that, if $N > k$, then $V_{\max}(i, T, N)$ is $+\infty$, i.e. that N conspirators can achieve the value $+\infty$. But by the above comments the number of conspirators needed to achieve the value $+\infty$ equals the number of elements of B less than $+\infty$, i.e. the number of elements of B. Thus k conspirators can achieve the value $+\infty$ and, if $N > k$, then N conspirators can also achieve the value $+\infty$. □

The bounds multiset theorem implies that for a non-leaf min node i, the length of the bounds sequence $\uparrow$bounds(i, T) is the sum of the length of the bounds sequences of its successor nodes. Thus the length of the sequence $\uparrow$bounds(i, T) is minimized at max nodes and summed at min nodes. The experiments described in Section 8 indicate that the bound sequences tend to remain short.

5. A Tree-Growth Procedure

This section presents a tree-growth procedure for determining the value of the root node. The tree-growth procedure acts on any well-formed subtree T of the game G. The procedure iteratively chooses leaf nodes for expansion. The tree should be grown in a way that restricts the set of likely root values. The growth procedure takes two parameters: a range parameter Δ and a conspiracy threshold N_t. The range parameter Δ states the accuracy to which the root value is to be determined. The conspiracy threshold N_t tells the procedure that conspiracies of more than N_t nodes are unlikely. The procedure terminates when the range of likely root values $[V_{\min}, V_{\max}]$ for the given threshold N_t is such that $V_{\max} - V_{\min}$ is less or equal to Δ. The size of the tree grown by the procedure is governed by the desired accuracy Δ and the conspiracy threshold N_t; the greater the desired accuracy, and the greater the required conspiracy threshold, the larger the resulting tree.

In determining the root value of a search tree one must be able to rule out

certain values. Intuitively ruling out a large value involves finding a strategy the min player can use for avoiding that value. Thus, when attempting to rule out a large value V one should expand leaf nodes which are involved in strategies the min player can use to avoid V; such nodes are called min strategy leaf nodes. A min strategy leaf node for avoiding V is defined here to be a node involved in a minimal conspiracy set for increasing the root value to V.

Definition 5.1. A leaf node k of T will be called a *min strategy leaf node* for avoiding a value V larger than the root minimax value if k is contained in some minimal conspiracy set for converting the root value of T to V.

Similarly, a leaf node j of T will be called a *max strategy leaf node* for avoiding a value V smaller than the root minimax value of T if j is contained in some minimal conspiracy set for converting the root value to V.

There is a technical reason for the term "min strategy leaf node". Technically, a min strategy is a plan of action for the min player: a min strategy is a subtree T' of T which includes the root node of T and such that if i is a non-leaf min node of T which is in T' then exactly one successor of i is in T' (the planned move) and if i is a non-leaf max node of T which is in T' then every successor of i is in T'. Intuitively, a min strategy establishes an upper bound on the root value; a min strategy is a plan of action for the min player and the root value can be no larger than the value achieved by this particular plan of action for the min player. For any given min strategy the max player will make moves that lead to the largest leaf value. Thus the value achieved by a particular min strategy is the maximum of the strategy leaf values. If the minimax value of T is V then there exists a min strategy whose maximum value is V and there also exists a max strategy (the dual notion) whose minimum value is V. Definition 5.1 uses the term "min strategy leaf node" because every min strategy leaf node for avoiding V is a leaf node in a min strategy for T whose maximum value is less than V (the proof is omitted here). The converse does not hold; there are leaf nodes in min strategies with maximum values less than V which are not min strategy leaf nodes for avoiding V under the above definition. The min strategy leaf nodes for avoiding V are the "shallow" leaf nodes in min strategies.

To choose a leaf node for expansion one must decide which of the values $V_{\min}$ or $V_{\max}$ to try to rule out. The tree-growth procedure tries to rule out the value which is farthest from the current minimax root value. When trying to rule out $V_{\max}$ one should expand a min strategy leaf node; when trying to rule out $V_{\min}$ one should expand a max strategy leaf node.

Basic tree-growth procedure. To determine a value for a node A in a min-max game G do the following:

Step 1. Initialize T to be the subtree of G which contains only the node A.

Step 2. If $V_{max}(A, T, N_t) - V_{min}(A, T, N_t) \leq \Delta$, then terminate and return the range $[V_{min}(A, T, N_t), V_{max}(A, T, N_t)]$.

Step 3. If $V_{max}(A, T, N_t)$ is further from the minimax value of the root node A than is $V_{min}(A, T, N_t)$, then enlarge the tree T by expanding the leftmost min strategy leaf node for avoiding $V_{max}(A, T, N_t)$, otherwise enlarge T by expanding the leftmost max strategy leaf node for avoiding $V_{min}(A, T, N_t)$.

Step 4. Go to Step 2.

The above procedure does not specify how to find min and max strategy leaf nodes. Fortunately there is a simple technique for finding such leaf nodes by starting at the root node and recursively descending the tree choosing successor nodes on the bases of the bounds sequences at those nodes. The following procedure returns a min strategy leaf node for avoiding V. The procedure for finding max strategy leaf nodes is the dual of that for finding min strategy leaf nodes.

Procedure MIN-STRATEGY-LEAF(i, V).

Step 1. If i is a leaf node return i.

Step 2. If i is a max node, then any minimal conspiracy set for increasing i to V will affect only a single successor node of i. In this case return the value of MIN-STRATEGY-LEAF(j, V) where j is any successor of i which minimizes $\uparrow$needed(j, T, V), i.e. such that there is no other successor j' with $\uparrow$needed(j', T, V) less than $\uparrow$needed(j, T, V).

Step 3. If i is a min node, then any minimal conspiracy set for increasing i to V must affect every successor of i whose minimax value is less than V. In this case return the value of MIN-STRATEGY-LEAF(j, V) where j is any successor of i with minimax value less than V.

The above procedure is nondeterministic; the choice of the successor node in Steps 2 and 3 are only partially constrained. The procedure always returns a min strategy leaf node for avoiding V and every min strategy leaf node for avoiding V is a possible value of the procedure (the proof is based on the comments in Steps 2 and 3). To find the leftmost min strategy leaf node one simply takes the leftmost possible successor at Steps 2 and 3. The condition specified in Step 3 ensures that the node returned is a member of a min strategy whose maximum value is less than V. The condition specified in Step 2 ensures that the node returned is "shallow".

6. An Example of Tree Growth

Consider the simple 1-ply tree shown in Fig. 3. The static value of the root node A is shown in parentheses next to the minimax value. The static values of non-leaf nodes will be important for understanding the way the tree has been grown.

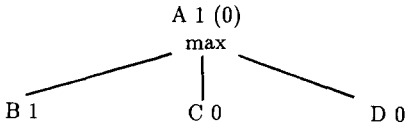

Fig. 3. A 1-ply tree.

Node A of Fig. 3 might represent a chess position in which the max (white) player has three moves. Furthermore the static values can be interpreted as piece counts: the static value is computed by multiplying each piece by its point value and summing over all pieces on the board. The static values assigned to the nodes B, C, and D indicate that one of the moves available to the max player captures a pawn (worth 1 point) while the two other moves do not capture pieces. In this game static values will be integers and the range parameter Δ will be zero; the procedure will not terminate until an exact root value is found.

In this example the nodes of the game G are sorted left-to-right in a best-first fashion based on the static values; if i is a max node, then successors of i with high static values are left of successors of i with low static values; if i is a min node, then successors with low static values are left of successors with high static values.

The tree-growth procedure can be used to enlarge the tree shown in Fig. 1. The first step is to compute the range of likely root values for this tree. In this example we will assume that the conspiracy threshold parameter N_t of the search procedure is two; if more than two conspirators are required to change the root value to V, then V is not likely. For the tree shown in Fig. 1 three conspirators are required to decrease the root value below 0. Thus $V_{\min}(A, T, 2)$ equals 0. However any one of the leaf nodes could act alone to increase the root value to $+\infty$. Thus $V_{\max}(A, T, 2)$ equals $+\infty$. Thus the range of likely root values for a conspiracy threshold of two is $[0, +\infty]$.

The likely root value which is farthest from the current minimax root value is clearly $+\infty$. Thus the tree-growth procedure will attempt to rule out the value $+\infty$ by expanding min strategies for avoiding $+\infty$. Each of the nodes B, C, and D is a min strategy leaf node. The tree-growth procedure expands the leftmost of these nodes to get the tree shown in Fig. 4.

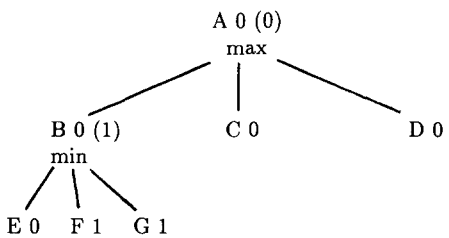

Fig. 4. The tree after one expansion.

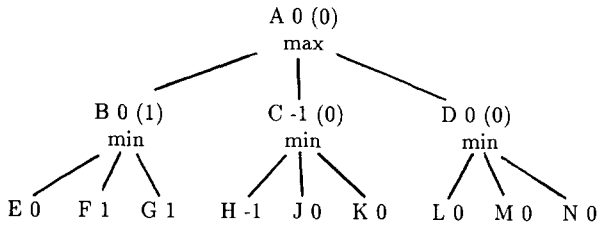

Fig. 5. The tree after three expansions.

For a conspiracy threshold of 2 the range of likely root values is still $[0, +\infty]$. The value $+\infty$ is still farthest from the current minimax root value so the procedure will enlarge the tree by expanding strategies to avoid $+\infty$. Either of the nodes C or D could act alone to increase the root value to $+\infty$. Thus both of these nodes are min strategy leaf nodes. The leftmost min strategy leaf node for the tree shown in Fig. 4 is the node C. After C is expanded the only min strategy leaf node for avoiding the value $+\infty$ will be the node D. Thus the next two nodes expanded by the tree-growth procedure are the nodes C and D. The result of expanding these nodes is shown in Fig. 5.

To compute the range of likely root values for the tree shown in Fig. 5 note that E can act alone to increase the root value to 1 but three conspirators are needed to increase the root value to 2. Furthermore E and L can conspire to decrease the root value to -1 but three conspirators are needed for the value -2. Thus the range of likely root values for this tree is $[-1, 1]$. In this case $V_{\min}$ and $V_{\max}$ are equally far from the root value 0. The tree-growth procedure given in the previous section breaks such ties in favor of $V_{\min}$. Thus the tree-growth procedure will now grow a max strategy leaf nodes in an attempt to rule out the value -1. The leftmost max strategy leaf node is the node E ($\{E, L\}$ is a minimal set of conspirators for decreasing the root value to -1). The result of expanding the node E is shown in Fig. 6.

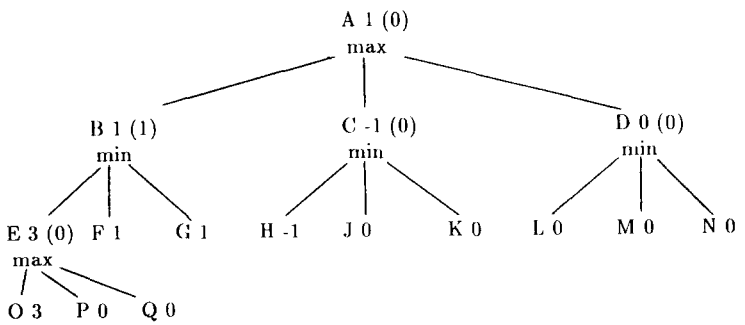

Fig. 6. The tree after four expansions.

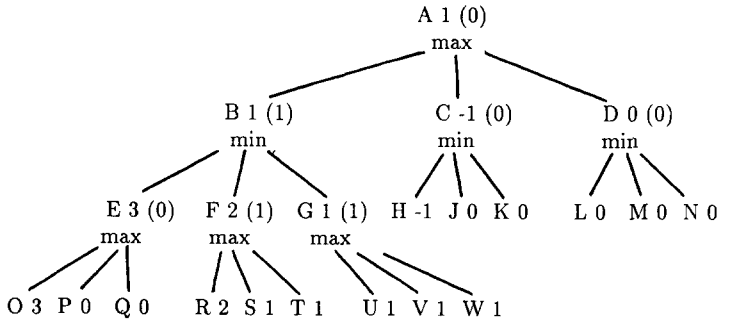

Fig. 7. The tree after six expansions.

Nodes F and G can conspire to convert the root value of Fig. 6 to 3; the range of likely root values for Fig. 6 is $[-1, 3]$. Since these values are equally far from the current root value the procedure defaults to ruling out the value -1 by expanding a max strategy leaf node. The leftmost max strategy leaf node is now F. After expanding F the leftmost max strategy leaf node will be G. The result of expanding both F and G is shown in Fig. 7.

For the tree shown in Fig. 7 three conspirators are needed to decrease the value of node B to -1. Thus four conspirators are needed to decrease the root value to -1. However the node O can act alone to decrease the root value to 0. The range of likely root values for the tree of Fig. 7 is $[0, 3]$. Since 3 is further from the current root value than 0 the tree-growth procedure now attempts to rule out the value 3 by expanding a min strategy leaf node. The nodes R, S and T are all min strategy leaf nodes and the value 3 is not eliminated until all three of these nodes have been expanded. The result of expanding these nodes is shown in Fig. 8. Figure 8 shows only the non-leaf nodes of the tree. For each node which has a leaf node as a successor the relevant Δ-needed parameter for that node is shown.

For the tree of Fig. 8 three conspirators are needed to increase the value of node F to 3. Thus four conspirators are needed to increase the value of B to 3, so the value 3 has been ruled out. The range of likely root values for the tree of Fig. 8 is $[0, 2]$. The tree-growth procedure now attempts to rule out the value 0 by expanding the leftmost max strategy leaf node. The node O (which is beneath node E) can act alone to decrease the value of node B to 0, thus decreasing the root value to 0. When the node O is expanded the root value does in fact drop to 0. After the root value has dropped to 0 the tree-growth procedure attempts to rule out the value 2 by expanding min strategy leaf nodes. The expansion of min strategy leaf nodes continues until the procedure terminates with the tree shown in Fig. 9. Figure 9, like Fig. 8, only shows non-leaf nodes.

The tree of Fig. 9 can be interpreted as a "proof" that the root minimax

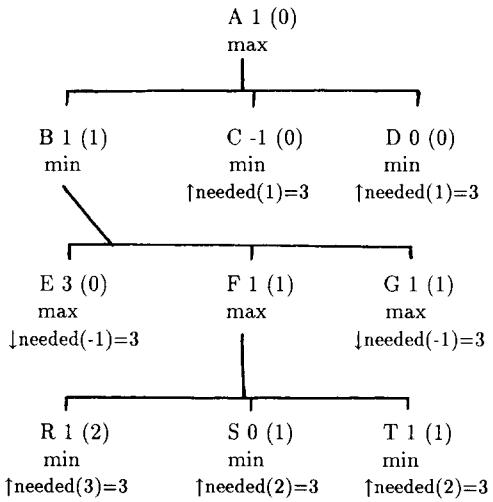

Fig. 8. The non-leaf nodes after nine expansions.

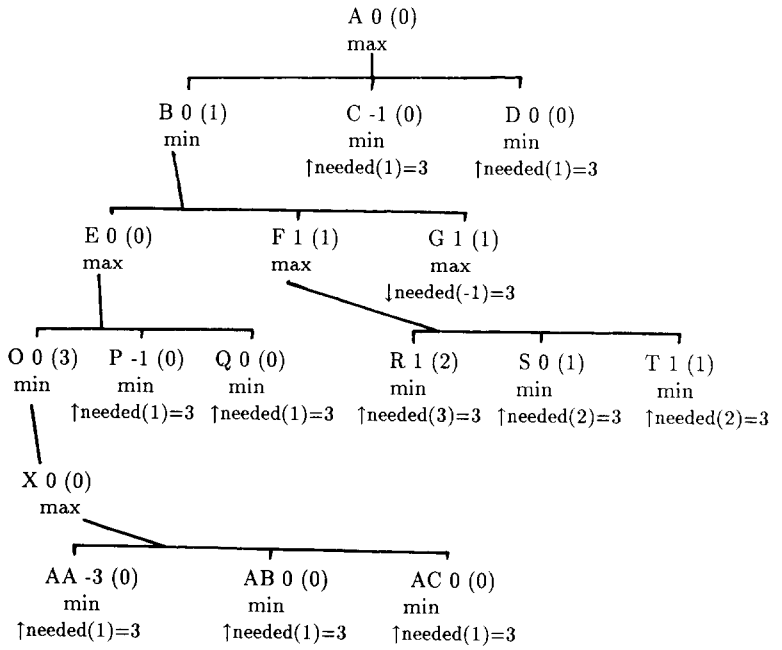

Fig. 9. The non-leaf nodes of the final tree (sixteen expansions).

value is 0; three conspirators are required to increase the root value to 1 and three conspirators are required to decrease the root value to -1. The tree can also be interpreted as providing a strategy that the min player can use to avoid the value 1 and a strategy that the max player can use to avoid the value -1. For example consider the strategy which the max player can use to avoid the value -1. From the initial position the max player has a choice of two moves each of which avoid the value -1. Suppose the max chooses node B. No matter how the min player responds at node B the max player has at least two choices for avoiding the value -1. Thus a total of three leaf nodes must conspire against the max player before the max player can no longer avoid the value -1. The strategy which the min player can use to avoid the value 1 is more involved. If the max player moves to either node C or D, then the min player immediately has three choices all of which avoid the value 1. If the max player moves to node B, then in order to avoid the value 1 the min player must move to node E. From node E the max player can choose between nodes O, P, or Q. If the max player chooses either P or Q, then the min player immediately has three choices for avoiding the value 1. If the max player chooses node O, then the min player is forced to move to node X. No matter how the max player moves from node X the min player immediately has three choices for avoiding the value 1.

The sequence of nodes leading from the root A to the node X form an exchange sequence. The max player begins by gaining a point (capturing a pawn). To avoid loosing material, the min player is forced to respond in a way that regains the point. The max player then gains three points (capturing a piece). To avoid a permanent loss of material, the min player is again forced to immediately recapture. By choosing capture moves the max player can "force" the min player to node X. The single node X can act alone to increase the root value to 1. To rule out the value 1 as a likely root value the value 1 must be ruled out as a likely value for the node X. However there are no capture moves available from node X so a 2-ply expansion of node X rules out 1 as a likely value.

Figure 9 shows that the three-growth procedure produces trees which are quite different from that produced by d-ply search with α-β pruning. The conspiracy-based procedure carries out capture moves to greater depths than non-capture moves. Capture moves generally have the property that there is only a small number of viable responses; the second player must immediately recapture the lost material. A move with a small number of viable responses will be explored more deeply than moves which leave the opponent many viable responses.

7. A Comparison with α-β Pruning

If all nodes in the game G have the same static value, then the tree-growth procedure of Section 5 is essentially the same as d-ply search with α-β pruning.

For example consider an infinite game G where every node in G has two successors and the static value of every node is 0. Now suppose this tree is expanded using the tree-growth procedure of Section 5 with a conspiracy threshold parameter N_t equal to three (if all static values are the same, the range parameter Δ is irrelevant). The initial range of likely root values is $[-\infty, +\infty]$ and the procedure defaults to attempting to rule out V_{min} by expanding max strategy leaf nodes. By repeatedly expanding the leftmost max strategy leaf node for avoiding the value $-\infty$ the procedure reaches a point where it has grown the tree shown in Fig. 10. Figure 10 does not show any minimax values; all minimax values are 0.

Any set of conspirators for converting the root minimax value of the tree in Fig. 10 to $-\infty$ must include nodes under both B and C. A max strategy leaf node is a member of a conspiracy set for decreasing the root value. Since any such conspiracy set must include a node under B, the leftmost max strategy leaf node must always be under node B. Similarly, any set of conspirators for decreasing the value of the max node D must include a node under both F and G. Thus the leftmost conspirator for decreasing node D must be under node F rather than G. Similarly, any leftmost conspirator for decreasing the value of max node E must be under node H rather than I. As the growth process continues it will grow the tree under the leftmost successor of each nonterminal max node ignoring the other successors of max nodes. Eventually the tree-growth procedure generates the tree shown in Fig. 11. At this point the root value $-\infty$ has been ruled out; four conspirators are needed to decrease the root value to anything below 0 so, at a conspiracy threshold of 3, the range of likely root values is $[0, +\infty]$. Figure 11 shows only the non-leaf nodes of the tree.

The nodes shown in Fig. 11 form a 4-ply max strategy. In general a d-ply max strategy is defined as follows:

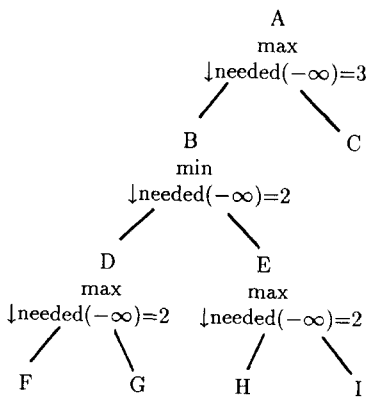

Fig. 10. A uniform tree after four expansions.

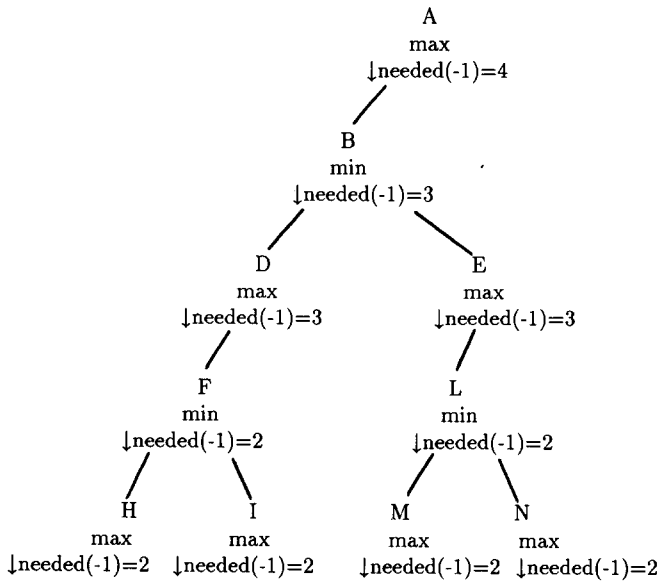

Fig. 11. A max strategy tree after ten expansions.

Definition 7.1. A strategy for the max player provides exactly one max player response to every min player option. More specifically let d be any natural number greater than 1. A *d-ply max strategy in the game G* is a subtree T_{max} of G satisfying the following conditions:

(1) Every node of T is within d ply of the root node of T.
(2) For every max node j of T which is less than d ply from the root of T exactly one successor node of j is in T.
(3) For every min node j in T which is less than d ply from the root of T every successor node of j is in T.

A d-ply max strategy T will be called *leftmost* if the successor chosen in condition (2) is always the leftmost successor.

Strategies for the min player are similarly defined: strategies for the min player specify exactly one min-player response to every max-player option.

Now consider standard d-ply search with α-β pruning. The search tree produced by d-ply search with α-β pruning always contains a d-ply max strategy T_{max} and a d-ply min strategy T_{min} such that T_{max} and T_{min} together constitute a "proof" that the minimax root value of the full d-ply search is a particular value V_A. The max strategy T_{max} constitutes a proof that the max player can avoid any value less than the root value V_A and the min strategy T_{min} constitutes a proof that the min player can avoid any value greater than

the root value V_A. An *optimal* d-ply α-β tree is one which can be written as the union of $T_{\max}$ and $T_{\min}$; an optimal tree contains just enough to prove that the root value of the full d-ply search is V_A. If all nodes have the same static value and the same branching factor, then the conspiracy search technique of Section 5 behaves like α-β search; both procedures grow a tree which can be written as the union of a d-ply min strategy and a d-ply max strategy. For the conspiracy-based procedure the depths of the max and min strategies, $d_{\min}$ and $d_{\max}$ respectively, are determined by the conspiracy threshold parameter N_t and the branching factor b as follows:

$$d_{\max} = 2\left\lfloor \frac{N_t - 1}{b - 1} \right\rfloor,$$

$$d_{\min} = d_{\max} + 1.$$

8. Empirical Results

An iterative deepening version of the tree-growth procedure of Section 5 has been tested on randomly generated games trees and compared with iteratively deepened α-β search. The tests indicate that conspiracy-based search achieves more accurate minimax values for the same number of nodes searched.

Experiments have shown that the following iterative deepening version of the tree-growth procedure works better than the basic procedure given in Section 5. The following procedure starts with a conspiracy threshold of 1 and gradually increases the conspiracy threshold as the root value becomes determined (to within Δ) for each conspiracy threshold.

Iteratively deepened tree-growth procedure.

Step 1. Initialize T to be the subtree of G which contains only the root node A.

Step 2. Let N be the least integer such that

$$V_{\max}(A, T, N) - V_{\min}(A, T, N) > \Delta.$$

Step 3. If $V_{\max}(A, T, N)$ is further from the minimax value of the root node A than is $V_{\min}(A, T, N)$, then enlarge the tree T by expanding the leftmost min strategy leaf node for avoiding $V_{\max}(A, T, N)$, otherwise enlarge T by expanding the leftmost max strategy leaf node for avoiding $V_{\min}(A, T, N)$.

Step 4. Go to Step 2.

The performance of the above tree-growth procedure was evaluated by measuring the error in the minimax value as a function of the number of nodes searched.

Definition 8.1. The *error* of a partial search tree T is the absolute value of the difference between the minimax value of the root node of T and the true value of that node in the underlying game.

The game trees used in the experiments had a fixed branching factor of ten and a depth of six ply. The static value of the root node of each tree was set to 0. If j was a successor node of node i, then the static value of j was set equal to the static value of i plus a randomly selected integer in the range $[-n, n]$. The parameter n, which determined the random variation in successor nodes, decreased as nodes got deeper in the tree. More specifically n equaled $7 - d$, where d is number of ply between the root node and the node whose static value is being assigned. Successors of the root node where assigned a static value with n equal to 6 while leaf nodes where assigned static values with n equal to 1.

Forty random game trees were generated according to the above scheme. For each game tree the exact minimax root value was determined by iteratively deepened α-β search to the full depth of the game. The above iterative deepening tree-growth procedure, with Δ equal to 0, was also applied to each game until the search tree contained 10,000 nodes. Data was taken at various points during both the iteratively deepened α-β search and the iteratively deepened conspiracy search. More specifically, when the number of nodes in the search tree reached certain thresholds the minimax value of the partial search tree was recorded. Given the minimax values of the partial trees at various points in the search, and the ultimate minimax value of the position, it was possible to compute the error in the root value of the partial trees at various points in the search. Table 1 shows the mean error for the forty games searched as a function of the number of nodes examined for both iteratively deepened α-β search and the above tree-growth procedure based on conspiracy numbers. The table shows that the mean error drops considerably faster for the above procedure than it does for iteratively deepened α-β.

Table 1
Mean error as a function of nodes searched

Search strategy	Mean error				
α-β search	1.45	1.80	0.80	0.95	0
Conspiracy search	1.52	0.95	0.15	0.02	0
Number of nodes searched	100	300	1,000	3,000	10,000

9. Determining a Best Move

The tree-growth procedure of Section 5 is designed to determine the value of a given node. In game playing, however, the goal is to determine an optimal

successor node. For a given node i in a search tree T one can define a best successor as follows:

Definition 9.1. Let i be a node in a search tree T. If i is a max node, then a *best successor* of i under N-way conspiracies is any successor j such that $V_{\min}(j, T, N)$ is greater than or equal to $V_{\max}(j', T, N)$ for all successor nodes j' other than j. If i is a min node, then a best successor for i under N-way conspiracies is defined in a dual way.

If j is a best successor of i, then the minimax value of i equals the minimax value of j and, furthermore, this will remain true under any N-way conspiracy, i.e. this will remain true under any modification of N leaf nodes. Thus, assuming conspiracies of more than N nodes are unlikely, a best successor node under N-way conspiracies is likely to be an optimal successor under the actual node values. This notion of an optimal successor is derived from the observations made by Berliner [7] in developing the B* algorithm.

Note that it is possible for a node i to have a best successor for N-way conspiracies even if the value of i has not been determined for N-way conspiracies, i.e. even when $V_{\max}(i, T, N)$ is strictly greater than $V_{\min}(i, T, N)$. Thus it is possible to determine a best successor even when exact minimax values have not been (or cannot be) determined.

The goal of a game playing program is to determine the best successor for a certain position. To determine the best successor of a max node one can either try a *prove-best* strategy by trying to raise the lower bound of a potentially best successor, or one can try a *disprove-rest* strategy by trying to lower the upper bound of all successors other than the potentially best successor. These observations are also derived from Berliner's B* algorithm. Under either strategy the first step is to pick a potential best successor j. Under the prove-best strategy (for a max root node) one should expand a max strategy leave node under j for avoiding $V_{\min}(j, T, N)$. Under the disprove-rest strategy one should pick some other successor j' such that $V_{\max}(j', T, N)$ is greater than $V_{\min}(j, T, N)$ and then expand a min strategy leaf node under j' for avoiding the value $V_{\max}(j', T, N)$. These observations can be incorporated into the following iteratively deepened procedure for finding a best successor:

Procedure to find a best successor.

Step 1. Initialize T to be the subtree of G which contains only a root node and its immediate successors.

Step 2. Let N be the least integer such that the root node has no best successor for N-way conspiracies (N will be at least 1).

Step 3. Let j be a best successor of the root node for $(N-1)$-way conspiracies.

Step 4. Choose a strategy: either prove-best or disprove-rest.

Step 5. If the strategy is prove-best, then expand the leftmost max strategy leaf node under j for avoiding the value $V_{\min}(j, T, N)$.

Step 6. If the strategy is disprove-rest, then choose some successor j' of the root node such that $V_{\max}(j', T, N)$ is greater than $V_{\min}(j, T, N)$ and expand a min strategy leaf node under j' for avoiding the value $V_{\max}(j', T, N)$.

Step 7. Go to Step 2.

The above procedure does not specify how to choose a strategy or how to choose an alternative successor in the disprove-rest strategy. With conspiracy numbers there is a lot of information that can be used in making these decisions. For example, the likelihood of decreasing $V_{\max}(j', T, N)$ below $V_{\min}(j, T, N)$ may depend on whether or not $V_{\max}(j', T, N-1)$ is greater than $V_{\min}(j, T, N)$. Further research is needed to find good heuristics for making these choices. Palay [8] has explored heuristics for making these choices in the context of B*.

10. Modifying the Static Evaluator

The previous section shows how ideas from the B* search technique can be used with conspiracy numbers. This section presents another idea for B* search that can be used with conspiracy numbers: the static evaluator can be modified to return ranges of values. Section 4 shows that all the relevant parameters of a node i in a tree T are determined by its bounds sequences $\uparrow\text{bounds}(i, T)$ and $\downarrow\text{bounds}(i, T)$. These bounds sequences are both singleton sequences for leaf nodes. The static evaluator could be modified to return bounds sequences which were longer than singletons and the procedures presented above could run without any additional modification.

A classical static evaluator could be used to generate more than singleton bounds. For example, consider a max leaf mode i with static value V. One could assign i an upper bound sequence of $\langle V, V + \Delta \rangle$ and a lower bound sequence of $\langle V, V, V - \Delta \rangle$ where Δ is a game-dependent parameter. Note that the lower bound sequence gives tighter bounds than the upper bound sequence: max nodes are more likely to increase under expansion than decrease.

In a particular game there may be ways of computing bound sequences by examining the structure of a given position. For example, "static" values in chess can be computed by performing an α-β minimax search of capture moves. Consider a max leaf node. Let V be the value returned by an α-β search of capture moves. Let V_{threat} be the value returned by an α-β search of capture moves *under the assumption that the min player moves first.* The value V_{threat} gives the threat presented by the min player if the max player does nothing. V_{threat} is a good lower bound on the value of the position. In this case the upper bound sequence for i can be assigned the singleton $\langle V \rangle$ and the lower bound sequence can be assigned the doubleton $\langle V, V_{\text{threat}} \rangle$.

11. Conclusions

Conspiracy numbers have several advantages over classical α-β search techniques. First, critical lines of play can be explored more deeply than noncritical lines. The outcome of a critical line of play can have a strong influence on the minimax value of the root node; in order to accurately determine the minimax root value one must accurately determine the outcome of critical lines of play. Second, conspiracy numbers provide a principled way of spending more time analyzing difficult positions than easy positions; one can continue searching until a given conspiracy threshold is reached. And third, because conspiracy numbers assign value ranges to nodes in the search tree, conspiracy numbers can be used to determine a best move in a manner analogous to that used in B*.

Ultimately the ideas presented here should be judged at tournaments for game playing machines.

ACKNOWLEDGMENT

Calvin Hsia inspired me to work on the ideas presented here by offering to write a chess program if I designed an algorithm (unfortunately the algorithm took longer than expected). Dana Nau's work on pathological games led me to think hard about the reasons that deeper searches are better. Ron Rivest and Hans Berliner both provided useful comments on early drafts of this paper. Jonathan Schaeffer provided valuable feedback by implementing the ideas presented here in a tactical analysis program for chess (his results will be reported elsewhere). Schaeffer's feedback led me to simplify the search procedure presented in Section 5 and pointed out the importance of iterative deepening.

REFERENCES

1. Nau, D.S., Decision quality as a function of search depth on game trees, *J. ACM* **30** (4) (1983) 687–708.
2. Nau, D.S., Pathology on game trees revisited and an alternative to Minimaxing, *Artificial Intelligence* **21** (1983) 221–244.
3. Berliner, H.J., An examination of brute force intelligence, in: *Proceedings IJCAI-81*, Vancouver, BC (1981).
4. Knuth, D.E. and Moore, R.W., An analysis of alpha-beta pruning, *Artificial Intelligence* **6** (1975) 293–326.
5. Stockman, G., A minimax algorithm better than alpha-beta? *Artificial Intelligence* **12** (1979) 179–196.
6. Roizen, I. and Pearl, J., A minimax algorithm better than alpha-beta? Yes and no, *Artificial Intelligence* **21** (1983) 199–220.
7. Berliner, H.J., The B* tree search algorithm: A best-first proof procedure, *Artificial Intelligence* **12** (1979) 23–40.
8. Palay, A.J., The B* tree search algorithm—New results, *Artificial Intelligence* **19** (1982) 145–163.

Received August 1985; revised version received August 1987