Alpha-Beta-Conspiracy Search
(Draft of Oct. 20, 1993)

David A. McAllester
Deniz Yuret

MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge Mass. 02139
dam@ai.mit.edu

Abstract: We introduce a variant of α-β search in which each node is associated with two depths rather than one. The purpose of α-β search is to find strategies for each player that together establish a value for the root position. A max strategy establishes a lower bound and the min strategy establishes an upper bound. It has long been observed that forced moves should be searched more deeply. Here we make the observation that in the max strategy we are only concerned with the forcedness of max moves and in the min strategy we are only concerned with the forcedness of min moves. This leads to two measures of depth — one for each strategy — and to a two depth variant of α-β called ABC search. The two depth approach can be formally derived from conspiracy theory and the structure of the ABC procedure is justified by two theorems relating ABC search and conspiracy numbers.

1 Introduction

Conspiracy numbers for min-max search were introduced in [McAllester, 1988]. Conspiracy numbers provide a game independent theoretical framework for guiding nonuniform growth in min-max game trees. The basic idea is to grow search trees for which one has confidence in the root min-max value. Conspiracy theory measures confidence by measuring the number of leaf values that would have to be changed to bring about a given change in the root value. This basic theoretical concept has been used with some success to solve tactical problems in chess [Schaeffer, 1990]. However, conspiracy theory has not been used directly in competitive computer chess programs. We believe this is due to a variety of deficiencies in the algorithm presented in [McAllester, 1988]. In this paper we present a new algorithm, α-β-conspiracy search, or ABC for short, which is based on conspiracy theory yet overcomes the deficiencies of the earlier procedure.

The algorithm presented here is a variant of classical α-β search. Like α-β it uses space that is linear in the depth of the search (but can be augmented with transposition tables). Like α-β search, ABC is a depth first procedure which terminates when depth bounds are exceeded. However, unlike classical α-β search, in the ABC procedure each node is associated with *two* depths — a depth of the node when viewed as an element of a max strategy and a second depth for the node when viewed as a element of a min strategy. The details of the ABC procedure described here can undoubtedly be improved in game specific ways. However, we have considerable confidence that the two depth approach will prove superior to the classical one depth approach in α-β search.

The ABC procedure overcomes four deficiencies of the procedure presented in [McAllester, 1988], which will call the naive conspiracy procedure, or NC for short. First, the NC procedure requires exponential space — the entire search tree must be stored. The ABC procedure, as a variant of α-β, requires only linear space. Second, the NC procedure is founded on the assumption that deviations from static values are statistically independent. This causes premature termination of the algorithm down important lines of play. The ABC procedure provides heuristic corrections for correlations. Third, the NC procedure involves considerably more overhead in expanding a single node than does classical α-β. The ABC procedure should achieve node evaluation rates roughly the same as classical α-β. Finally, the NC procedure does not allow for the incorporation of game specific extension heuristics such as check extensions. Although we argue that these game specific heuristics are just special case encodings of a general conspiracy theory principle, the direct implementation of these heuristics achieves greater efficiency than can be achieved using only static values. In section 10.3 we discuss a "static option estimator" for incorporating game specific heuristics such as check extensions.

ABC search uses conspiracy numbers to control search depth. The goal of α-β search is to construct strategies for each player that together establish a value for the root position. The max strategy establishes a lower bound and the min strategy establishes an upper bound. Conspiracy numbers can be used to measure the "safety" of these two strategies. A max strategy is safe to the extent that there exist options to the max moves selected in the strategy. If one of the lines of play in the strategy turns out not to work, i.e., not to achieve the desired root value, then a different line of play can be chosen provided that options exist to the selected moves in the strategy. A strategy is unsafe if there are no options to the selected moves, i.e., the selected moves are forced. For technical reasons described in section 4 we need to work with augmentations of strategies. The conspiracy depth of a strategy is the least number of leaf nodes in the augmentation of that strategy whose value must be changed in order to defeat the strategy. It is possible to assign a depth to each node in the strategy such that the conspiracy depth of the strategy is simply the depth of the shallowest leaf. Depth in a max strategy is called max depth and depth in a min strategy is called min depth. The max depth in a max strategy only increases at max moves — this is where the max player has options supporting the safety of the strategy. If a max move in a max strategy is forced, i.e., there are no options to that move, then max depth does not increase across the move and the search must be carried further to achieve a desired safety (depth measurement). If a min move happens to be forced in a max strategy this fact does not influence max depth — max depth is only measuring the options of the max player.

The idea that forced lines of play should be explored more deeply goes back to the original chess papers of Shanon and Turing [Shanon, 1950], [Turing *et al.*, 1953]. It is one of the justifications for the value of quiescence search in chess. It also clearly underlies the use of check extensions. It is also the motivation for the singular extension heuristic [Anantharaman *et al.*, 1990]. However, we believe that effective use of these heuristics has been frustrated by the use of a single depth measurement. Max depth extensions should be granted when the max move is forced, min depth extensions when the min move is forced, and the goal of the search should to be find a max strategy and a min strategy of sufficient max depth and min depth respectively.

2 Conspiracy Numbers

Consider a partially expanded min-max search tree. Every node in such a tree is classified as either a max node or a min node in such a way that the child of every max node is a min node and the child of every min node is a max node. The leaves of the tree are associated with static values. The min-max value of a leaf node is defined to be its static value; the min-max value of nonleaf max node is the maximum of the min-max value of its children; and the min-max value of a min node is defined in the dual way. Nonleaf nodes will be called internal nodes. Let δ be a number called the *singular margin*.[1] Conspiracy theory can be formulated using the following definition.

> **Definition:** Let T be a search tree with min-max value $V[T]$. The *lower bound conspiracy number* of T, denoted $C_<[T]$, is the number of leaf static values that must be changed to bring the root min-max value down to $V[T]-\delta$. The *upper bound conspiracy number* of T, denoted $C_>[T]$, is the number of leaves that must be changed to bring the leaf value up to $V[T]+\delta$.

Classical α-β search can be viewed as finding strategies for the two players — an optimally pruned uniform depth α-β tree is the union of a min strategy and a max strategy. By giving a particular plan of attack for the max player, the max strategy establishes a lower bound on the min-max root value for the given depth. By giving a plan of attack for the min player, the min strategy establishes an upper bound on the min-max root value. If these two bounds are the same then we have a proof that the min-max value of the root for the given depth is the value of the two strategies. $C_<[T]$ as defined as defined above expresses our confidence that the lower bound established by the max strategy will not be defeated by further expansion of the search

[1] The term "singular margin" comes from the singular extension algorithm [Anantharaman *et al.*, 1990]. Singular extensions are discussed in more detail in section 9.

tree. $C_>[T]$ expresses our confidence that the upper bound established by the min strategy will not be defeated.

Consider a uniform two ply search tree in which all leaf nodes have static value 0 and where the root node is a max node. In order to bring the root value up to δ all of the leaves that are children of the same min child of the root must be increased to δ. In order to bring the root value down to $-\delta$ one child of each min node must be brought down to $-\delta$. So $C_<[T]$ and $C_>[T]$ both equal the branching factor of the tree.

3 A Statistical Interpretation

It is possible to provide a statistical interpretation of conspiracy numbers. This is done by constructing a certain statistical model of the relationship between the static values of nodes and their true values. We assume that the true values are generated nondeterministically from the static values. Let ϵ be a number in the interval $[0, \frac{1}{2}]$. We assume that each leaf node has probability ϵ of having a true value equal to $+\infty$; probability ϵ of having true value $-\infty$; and probability $1 - 2\epsilon$ of having a true value equal to its static value. We let $G[T]$ be the tree in which the static values of the leaves of T are replaced (according to the probability distribution described) by their true values. We think of the true value tree, $G[T]$, as being nondeterministically generated from T.

This is an admittedly simplistic model. It assumes that the deviations of the true values from the static values are statistically independent at each leaf node. It also assumes that all deviations of true values from static values are either the value $+\infty$ or $-\infty$. However, this model seems to provide insight into the statistical significance of conspiracy numbers.

To state the precise relationship between this statistical model and conspiracy numbers we need some additional terminology. If $V[G[T]] \leq V[T] - \delta$ then we say that the max strategy in T failed. We let $P_r[F_{\max}|T]$ be the probability that the max strategy of T fails. We let $F_{\min}$ represent the event that $V[G[T]] \geq V[T] + \delta$, in which case we say that the min strategy failed, and define $P_r[F_{\min}|T]$ to be the probability of this event for the tree T. The following theorem states that both of these failure probabilities are polynomials in ϵ. Because ϵ is less than 1 we define the order of such polynomials to the be the *smallest* exponent of ϵ in a nonzero term. For example, $1 - \epsilon^2$ is order

0 while $4\epsilon + 6\epsilon^2$ is order 1.

Theorem: $P_r[F_{\max}|T]$ is a polynomial in ϵ of order $C_<[T]$ and $P_r[F_{\min}|T]$ is a polynomial in ϵ of order $C_>[T]$.

Proof: We consider only the lower bound conspiracy numbers — the proof for upper bound conspiracy numbers is similar. Let n be a node in the search tree; let $V[n, T]$ denote the min-max value of n as a member of the tree T; and let $C_<[n, T]$ be the number of leaf nodes that must be changed to bring the min-max value of n to a value less than or equal to $V[T] - \delta$. We will say that a node n *fails* (for the max player) in T if $V[n,\ G[T]] \leq V[T] - \delta$. The probability that a node n fails will be denoted as $P_r[F_{\max}[n]|T]$. We now prove by induction on the number of nodes below n in the search tree that $P_r[F_{\max}[n]|T]$ is a polynomial in ϵ of order $C_<[n,\ T]$. This will establish the desired result for the root node. First we consider a leaf node n. If the static value of n is greater than $V[T] - \delta$ then we have $C_<[n,\ T] = 1$ and $P_r[F_{\max}[n]|T] = \epsilon$ so the result holds. If the static value of n is less than or equal to $V[T] - \delta$ then we have $C_<[n,\ T] = 0$ and $P_r[F_{\max}[n]|T] = 1 - \epsilon$ so the result again holds. Now consider an internal max node n. Note that all children of n must fail before n fails. This implies

$$C_<[n,\ T] = \sum_c C_<[c, T]$$

where c ranges over the children of n. Since all children must fail before n fails, and since the children behave independently, we also have have

$$P_r[F_{\max}[n]|T] = \prod_c P_r[F_{\max}[c]|T].$$

From this equation we can see that $P_r[F_{\max}[n]|T]$ is a polynomial in ϵ and that the order of this polynomial is the sum of the orders of the polynomials for $P_r[F_{\max}[c]|T]$. Now by the induction hypothesis for the children nodes we have that the order of the polynomial for $P_r[F_{\max}[n]|T]$ is $C_<[n,\ T]$. Now suppose that n is a min node. A min node will fail if any one of its children fail. Hence we have

$$C_<[n,\ T] = \min_c C_<[c,\ T]$$

where c ranges over the children of n. The probability that n will fail can be written as

$$P_r[F_{\max}[n]|T] = 1 - \prod_c (1 - P_r[F_{\max}[c]|T]).$$

This implies that the order of the polynomial for $P_r[F_{\max}[n]|T]$ is the minimum of the order for the polynomials for $P_r[F_{\max}[c]|T]$. The result now follows from the induction hypothesis for the children nodes.

The statistical model underlying the above theorem is clearly unrealistic. However it does show that conspiracy numbers are a limiting case of more general statistical formulations such as that given by Baum and Smith [Baum and Smith, 1993]. Both the above model and the one described by Baum and Smith make the assumption that the deviation of true values from static values behaves independently at each leaf node. It seems, however, that the deviations of true values from static values do not behave independently in chess. For example suppose that one has failed to incorporate the importance of passed pawns into the static evaluator. In this case a position can be lost due to the presence of an unstoppable past pawn but the static evaluator does not recognize the danger. The pawn advancement may not happen for a great many moves. In this case virtually all of the static values in the critical lines of play are in error *in the same way* — they call positions even that are actually badly behind. Although most static evaluators do incorporate the concept of a passed pawn, there are undoubtedly many subtle strategic features of positions that are not incorporated into static evaluators and which cause errors in static values to be correlated. The next two sections describe ways of correcting for correlations.

4 Strategies

The ABC procedure attempts to overcome the correlations by assuming that errors in static values at different levels of the tree are less correlated than errors in static values at the same level. This seems plausible in chess. To require the conspiracies to involve different levels of the tree we focus on strategies rather than arbitrary trees. A strategy is a plan of action for one of the two players. This plan of action must take into account all possible options of the opponent. For example, a strategy for the max player specifies at each max node a particular planned move. At each min node a max strategy must include all the children of that node.

> **Definition:** A *max strategy* is a tree which includes exactly one child of every internal max node and includes all children of every internal min node. A *min strategy* is defined in the dual way.

Figure 1 shows a max strategy. Max nodes are represented by open circles and min nodes are represented by closed circles. The strategy shown in this figure happens to have uniform depth, but in general the above definition allows max strategies of nonuniform depth. Suppose all of the leaf nodes in this strategy have static value 0. In that case the strategy shows that there is a plan of action for the max player that can achieve the value 0 five ply down from the root. However, there may be other plans of action for the max player that achieve values larger than 0. In general, max strategies can be used to establish lower bounds on the d-ply min-max root value and min strategies establish upper bounds on the d-ply min-max root value. In very large games such as chess one is usually not particularly interested in the d-ply min-max value — all successful chess programs search to different depths down different lines of play. In very large trees where one must use static values rather than true values it is never possible to establish any true bounds on the root min max value. However, intuitively one still wants to think of max strategies as establishing lower bounds and min strategies establishing upper bounds. We want to find max strategies and min strategies that are "safe", i.e., such that we have confidence in the bounds established by those strategies.

As one can see from figure 1, the value of a max strategy is the minimum of the static values of its leaf nodes. Let $T_{\max}$ be a max strategy with value v. Since $T_{\max}$ is only one of many possible plans of action for the max player, we should think of v as a lower bound on the root value. One can try to measure the safety of the strategy by asking how likely it is that the value of the strategy would go down if the leaf values were replaced by their true value.

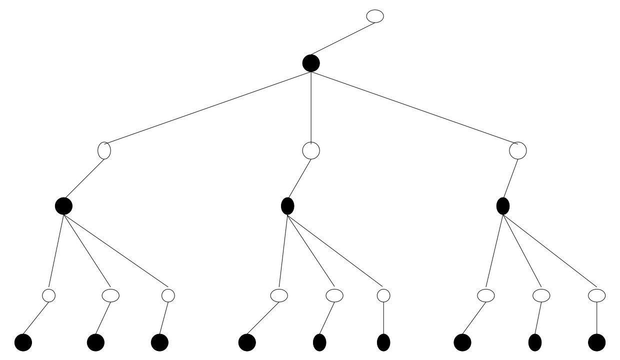


Figure 1: A max strategy

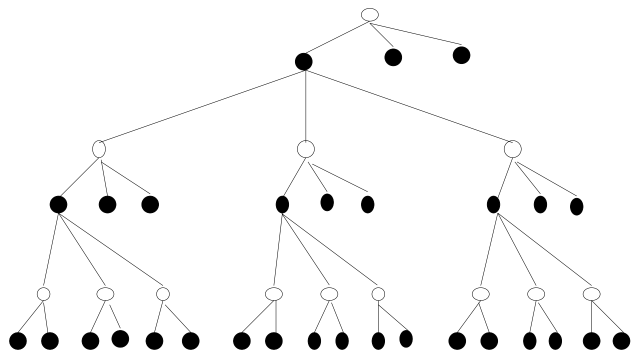


Figure 2: The augmentation of a max strategy

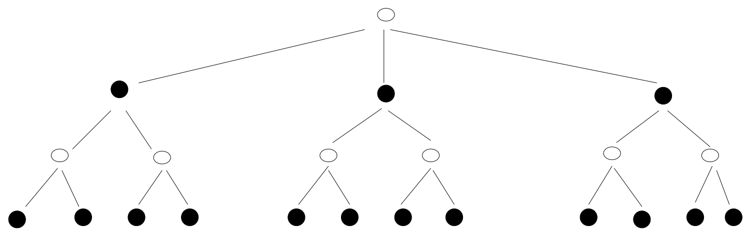


Figure 3: A uniform depth tree

This corresponds to asking how large a conspiracy is among the leaf nodes to bring the value of strategy down. Unfortunately, for any max strategy $T_{\max}$, the lower bound conspiracy number $C_<[T]$ is 1. We are not interested in $C_>[T]$ because $T_{\max}$ can not be used to justify any upper bound on the root value. Hence conspiracy numbers can not be used directly to measure the safety of strategies. To use conspiracy numbers to measure the safety of strategies we define the augmentation of a strategy.

> **Definition:** If $T_{\max}$ is a max strategy then the *augmentation* of $T_{\max}$, denoted $A[T]$, is defined to be $T_{\max}$ plus all children of all internal max nodes of $T_{\max}$. The augmentation of a min strategy is defined in the dual way.

Let $T_{\max}$ be the max strategy shown in figure 1. The augmentation, $A[T]$, is shown in figure 2. Suppose that all of the leaf nodes in $A[T]$ have value 0. In this case we have that $C_<[A[T]]$ is 6 — at least six leaf nodes in $A[T]$ must conspire to reduce the root value of $A[T]$. The concept of augmentation allows us to associate each strategy with a confidence measure.

> **Definition:** Let $T_{\max}$ be a max strategy with value v. We define $C_{\max}[T]$ to be the minimum number of leaf nodes in $A[T]$ that must be changed to bring the value of $A[T]$ down to $v - \delta$. For a min strategy $T_{\max}$ we define $C_{\min}[T]$ in the dual way.

For example, if $T_{\max}$ is the strategy shown in figure 1, and all leaf values in the augmentation $A[T]$ shown in figure 2 have value 0, then $C_{\max}[T]$ is 6. In general $A[T]$ contains more options for the max player than the "pure" strategy $T_{\max}$. Hence the min-max value of $A[T]$ can be larger than the min-max value of $T_{\max}$. However, we are interested in measuring our confidence in the min-max value of $T_{\max}$ as a lower bound on the root value so we require conspiracies that defeat the value of $T_{\max}$ (rather than the value of $A[T]$).

Again let $T_{\max}$ be the tree shown in figure 1. It is interesting to compare the tree $A[T]$ shown in figure 2 to the tree shown in figure 3 which we will call G. we assume that all the static values in $A[T]$ and G are 0. In this case we have that $C_<[G] = 6$ — each of the max moves at the root must be defeated to defeat the root value and defeating any one of these moves requires changing at least two leaf nodes. Even though $C_{\max}[T]$ and $C_<[G]$ are both 6, we believe that $A[T]$ provides a safer lower bound than the tree G in games like chess where nodes at the same level tend to have correlated errors in static values. Conspiracies for defeating G involve nodes that are all at the same level of the tree. On the other hand, conspiracies for defeating $A[T]$ must involve nodes from three different levels.

5 Conspiracy Depth

The conspiracy number $C_{\max}[T_{\max}]$ of a max strategy $T_{\max}$ can be interpreted as a measure of the "depth" of the strategy $T_{\max}$. In particular we have the following definition and lemma.

Definition: Let $T_{\max}$ be a max strategy with value v. For each max node n in $T_{\max}$ we define the *max options* for n in $T_{\max}$, denoted $O[n,\ T_{\max}]$, to be those children c of n that are not contained in the max strategy and have static value greater than $v-\delta$. Now let m be any node in the max strategy $T_{\max}$. We define the *conspiracy depth* of m in the max strategy $T_{\max}$, denoted $d_{\max}[m, T_{\max}]$, to be the sum over all max nodes n above m of $|O[n,\ T_{\max}]|$.

Lemma: $C_{\max}[T_{\max}]$ equals the minimum over all leaf nodes m in $T_{\max}$ of $d_{\max}[m,\ T_{\max}] + 1$.

Consider the two trees shown in figure 4. These are both augmented strategies. Let $A[T]$ be the left hand tree and let $A[G]$ be the right hand tree where T and G are the max strategies which select the leftmost branch at each max node. Assume that all leaf nodes have static value 0. We have that $C_{\max}[T] = C_{\max}[G] = 8$. We also have that each leaf node in the two

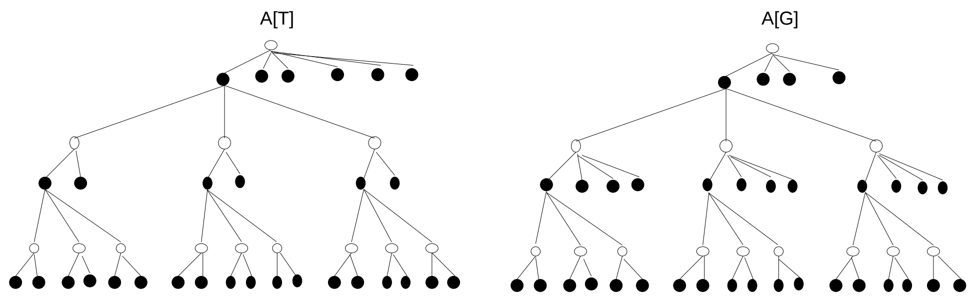


Figure 4: Two trees with equal conspiracy depth

strategies has conspiracy depth 7. The above lemma states that $C_{\max}[t]$ is always one greater than the depth of the shallowest leaf. Note that the leftmost child of the root position has depth 5 in the first strategy but only depth 3 in the second strategy.

6 Correcting for Correlations

Although augmentations of strategies seem safer than other tree shapes, we believe that we can give a measure of the safety of a max strategy $T_{\max}$ that is somewhat better than the conspiracy number $C_{\max}[T_{\max}]$. Again let $A[T]$ be the left hand tree and let $A[G]$ be the right hand tree in figure 4. Again assume that all leaf nodes have static value 0 so that $C_{\max}[T] = C_{\max}[G] = 8$. Note however, that conspiracies for defeating $T_{\max}$ involve five conspirators that are all at the same level while conspiracies for defeating G involve at most three conspirators from the same level. Since conspiracies among nodes at the same level seem more likely, T seems more likely to be defeated than G. We are really interested in the number of "independent conspirators" required to defeat a strategy. To approximate a count of the number of independent conspirators we place an upper limit on the number of conspirators that can come from a single level of the tree. We also normalize the conspiracy numbers so that the largest contribution to a conspiracy number from a single level of the tree is 1. Formally we introduce a monotone function S from the natural numbers to real numbers. The function S should have roughly the shape shown in figure 5 where b is the typical branching factor of the game. As shown in the figure, the function S should exhibit "diminishing returns"

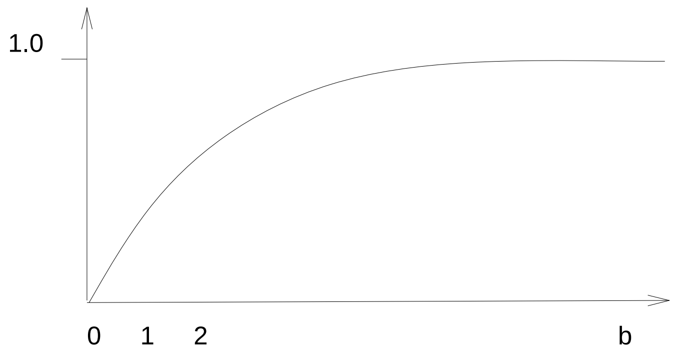


Figure 5: A typical function S

for additional options and should approach the value 1 for large option sets. Using the function S to bound the contribution to depth from any single move we define the concept of adjusted conspiracy depth as follows.

Definition: Let $T_{\max}$ be a max strategy with value v and let m be any node in the max strategy $T_{\max}$. We define the *adjusted conspiracy depth* of m in the max strategy $T_{\max}$, denoted $D_{\max}[m, T_{\max}]$, to be the sum over all max nodes n above m of $S(|O[n,\ T_{\max}]|)$. The adjusted conspiracy depth of $T_{\max}$, denoted $D_{\max}[T_{\max}]$, the minimum over all leaf nodes m in $T_{\max}$ of $D_{\max}[m,\ T_{\max}]$. For a node n in a min strategy $T_{\min}$ the numbers $D_{\min}[n,\ T_{\min}]$ and $D_{\min}[T_{\min}]$ are defined in an analogous way.

For example consider the two trees shown in figure 4 and as before let $A[T]$ be the tree on the left and $A[G]$ be the tree on the right. Assume that all static values are 0. The left most leaf node in T has an adjusted conspiracy depth of $S(1) + S(1) + S(5)$. All leaf nodes of T have the same adjusted conspiracy depth so $D_{\max}[T] = S(1) + S(1) + S(5)$. The leftmost leaf node in G has an adjusted conspiracy depth of $S(1) + S(3) + S(3)$. Reasonable values for $S(1)$, $S(3)$ and $S(5)$ might be .4, .9 and 1.0. Under these values we have that $D_{\max}[T] = 1.8$ while $D_{\max}[G] = 2.2$. The following lemma shows that, for different functions S, adjusted conspiracy depth can simulate either classical ply depth or (unadjusted) conspiracy depth.

Lemma: Let $T_{\max}$ be a max strategy whose root node is a max node. If $S(|O[m,\ T_{\max}]|) = 1$ for all max nodes in $T_{\max}$ then $D_{\max}[T_{\max}]$ is $\left\lfloor \frac{d+1}{2} \right\rfloor$ where d is the classical ply depth of the shallowest leaf node in $T_{\max}$. If $S(|O[m,\ T_{\max}]|) = \frac{|O[m,\, T_{\max}]|}{b}$ for all max nodes m in $T_{\max}$ then $D_{\max}[T_{\max}] = \frac{C_{\max}[T_{\max}]-1}{b}$.

Rather than derive a function S from statistical assumptions about correlations we simply propose a function which seems to have the appropriate heuristic properties. We suggest the following function S for use in chess where b is the typical branching factor of a chess position (approximately 30) and $\gamma \geq 1$ is a real valued constant which can be tuned empirically.

$$S(k) = \begin{cases} 1 & \text{if } k \geq b \\ 1 - \left(\frac{b-k}{b}\right)^{\gamma} & \text{otherwise} \end{cases}$$

For $\gamma = 1$ we get the case where $D_{\max}[T_{\max}] = \frac{C_{\max}[T_{\max}]-1}{b}$ and for $\gamma > b$, and $O[m,\ T_{\max}]$ nonempty for each max node m in $T_{\max}$, we have $D_{\max}[T_{\max}] \approx \left\lfloor \frac{d+1}{2} \right\rfloor$ where d is the classical ply depth of the shallowest leaf node in $T_{\max}$. So by tuning γ from 1 to b we can move continuously from conspiracy depth to classical ply depth.

7 The ABC Procedure

The ABC procedure presented here is essentially classical α-β search modified to use adjusted conspiracy depth rather than classical ply depth in determining when to terminate the search. The ABC procedure is shown in figure 6 and auxiliary procedures are shown in figure 7. The ABC procedure takes seven arguments. The first argument is the position being evaluated. The next two arguments are the α and β arguments of the classical α-β procedure. The next two arguments are two depth parameters — one specifying the desired depth of the max strategy and one specifying the desired depth of the min strategy. The next two arguments are lists of lists of values called option values. Each of these arguments is a list of lists of static values of siblings of the given node and its ancestors. The initial procedure call is

$$\text{ABC(root}, -\infty, +\infty, d_{\max}, d_{\min}, \textbf{nil}, \textbf{nil})$$

```
Procedure ABC(**node**, $\alpha$, $\beta$, $d_{\max}$, $d_{\min}$, **max-options**, **min-options**)
  1. If terminate?(**node**, $\alpha$, $\beta$, $d_{\max}$, $d_{\min}$, **max-options**, **min-options**) then
     return the static value of **node**.
  2. If **node** is a max node then
     (a) set $v$ to $\alpha$
     (b) for each child $c$ of **node**:
         i. let **sibling-options** be the list of static values of the siblings of $c$.
         ii. let **next-max-options** be the the list whose first element is the
             list **sibling-options** and whose remaining elements are the ele-
             ments of **max-options**.
         iii. set $v$ to max($v$, ABC($c$, $v$, $\beta$, $d_{\max}$, $d_{\min}$, **next-max-options**,
              **min-options**))
         iv. if $v \geq \beta$ return $v$ as the value of ABC.
     (c) return $v$.
  3. If **node** is a min node the do the dual of the max node case.
```

Figure 6: The ABC procedure

where **nil** represents the empty list. This top level call returns a value v with the guarantee that there exists a max strategy $T_{\max}$ and a min strategy $T_{\min}$ both of which have value v and such that $D_{\max}[T_{\max}] \geq d_{\max}$ and $D_{\min}[T_{\min}] \geq d_{\min}$. The procedure is identical to the classical α-β search procedure except for the termination test — the test to determine whether one should simply return the static value of the given position. Auxiliary procedures are shown in figure 7.

There are two fundamental properties to be established for the ABC procedure. First, if it returns the value v then there exist max and min strategies with the desired properties. Second, under optimal move ordering the search tree generated by the procedure is minimal in the sense that no proper subtree adequately establishes a value for the root. The proofs of the following theorems are given in an appendix.

Definition: A tree T is said to *contain* the max strategy $T_{\max}$ if

```
Procedure terminate?(node, $\alpha$, $\beta$, $d_{\max}$, $d_{\min}$, max-options, min-options)

    1. let $v$ the static value of node
    2. If $v \geq \beta$ and max-depth($\beta$, max-options) $\geq d_{\max}$ then return true (terminate).
    3. If $v \leq \alpha$ and min-depth($\alpha$, min-options) $\geq d_{\min}$ then return true (terminate).
    4. If $\alpha < v < \beta$ and max-depth($v$, max-options) $\geq d_{\max}$ and
       min-depth($v$, min-options) $\geq d_{\min}$ then return true (terminate).
    5. Otherwise return false (continue).

Procedure max-depth($v$, max-options)

    return the sum over all value lists $V$ in max-options of $S(k)$ where
    $k$ is the number of elements of $V$ greater than $v - \delta$.

Procedure min-depth($v$, min-options)

    return the sum over all value lists $V$ in min-options of $S(k)$ where $k$
    is the number of elements of $V$ less than $v + \delta$.
```

Figure 7: auxiliary procedures

the two trees have the same root and every leaf node of $T_{\max}$ is a leaf node of T. A similar definition applies for min strategies.

Definition: A tree T establishes a value v up to depth $<d_{\max},\ d_{\min}>$ if T contains a max strategy $T_{\max}$ and min strategy $T_{\min}$ both of which have value v and such that $D_{\max}[T_{\max}] \geq d_{\max}$ and $D_{\min}[T_{\min}] \geq d_{\min}$.

Theorem: If the top level call

$$\text{ABC}(\text{root},\ -\infty,\ +\infty,\ d_{\max},\ d_{\min},\ \textbf{nil},\ \textbf{nil})$$

returns value v then the tree of nodes examined by the search establishes v up to depth $<d_{\max},\ d_{\min}>$.

Definition: For each node n examined by a search we let v_n be the value computed for that node. A search is *optimally ordered* if at each internal node n we have that v_n equals v_c where c is the first child of n examined by the procedure.

Theorem: Let T be the tree of nodes examined by a search using depth parameters $<d_{\max}, d_{\min}>$. If the search is optimally ordered then no proper subtree of T establishes any value up to depth $<d_{\max}, d_{\min}>$.

The second theorem states that under optimal move ordering the procedure examines a minimal search tree. Since move ordering can be made nearly optimal in chess, the above theorem indicates that ABC should be nearly optimal for generating chess strategies of sufficient conspiracy depth. Move ordering is even more important for ABC than it is for classical α-β. In ABC search, move ordering determines both the amount of α-β pruning performed and the depth to which certain lines are searched. Optimal move ordering not only improves pruning but also allows the search to terminate at shallower levels. To see the effect on search depth note that the termination test can measure depth relative to α or β when the static value falls outside of the search window. In such cases the measuring the depth relative to α or β rather than the static value increases the likelihood of termination. Furthermore, the tighter the search window the more likely the termination and hence the shallower the search (as measured in classical ply depth). Optimal move ordering makes the search window as tight as possible on positions off the principle variation.

The interaction of the search window with the search depth also indicates that narrow window searches can be quite effective. Consider a top level call of the form

$$\text{ABC}(\text{root}, \alpha, \beta, d_{\max}, d_{\min}, \textbf{nil}, \textbf{nil})$$

which searches a tree T and returns value v. If $v > \alpha$ then T contains a max strategy $T_{\max}$ with value v and such $D_{\max}[T_{\max}] \geq d_{\max}$. If $v < \beta$ then T contains a min strategy $T_{\min}$ with value v and such that $D_{\min}[T_{\min}] \geq d_{\min}$. These two facts together provide a generalization of the first theorem above.

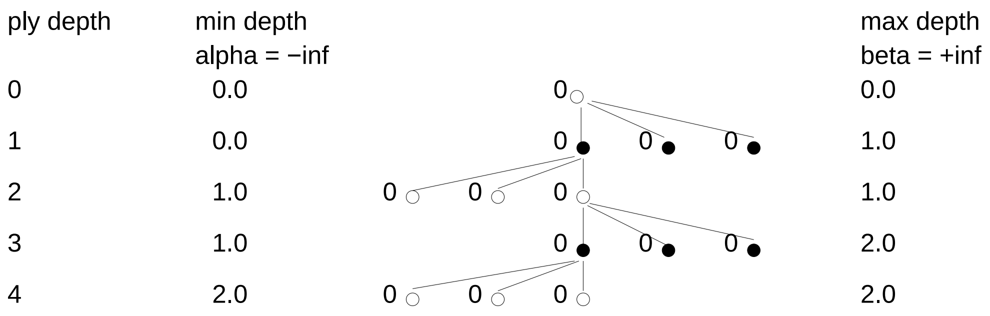


Figure 8: The nominal case

8 Some Examples

In this section we consider three hypothetical examples designed to illuminate aspects of the ABC procedure. The first involves discovering a deep combination. The second involves discovering a flaw in what initially appears to be a good attack. The third involves discovering a sacrifice that leads to a winning combination. These examples help to illustrate the relationship between the game independent notion of conspiracy depth and more familiar classical chess ideas.

In applying the ABC procedure to chess we believe that some form of classical quiescence search should be used as the static evaluator. Quiescence search in chess is an α-β search of a certain game tree that might be called the stand-pat or capture game. At each node the player to move has the option of stopping and taking the static value as the true value, the stand-pat option, or capturing a piece and letting the opponent move. If no captures are available the player must stand pat. In practice α-β search can be used to compute the exact value of the stand-pat or capture game tree rooted at any node. We will call this the quiescence value of the node.

In chess the quiescence value of a max position is usually close to the max of the quiescence values of its children with the dual statement holding at min nodes. This can not always be true because otherwise quiescence values would equal true values. The static values shown in the examples of this section are like quiescence values in that the static value of a max node is a good predictor of the max of the static values of its children. The reader should feel free to assume that these values are chess quiescence values.

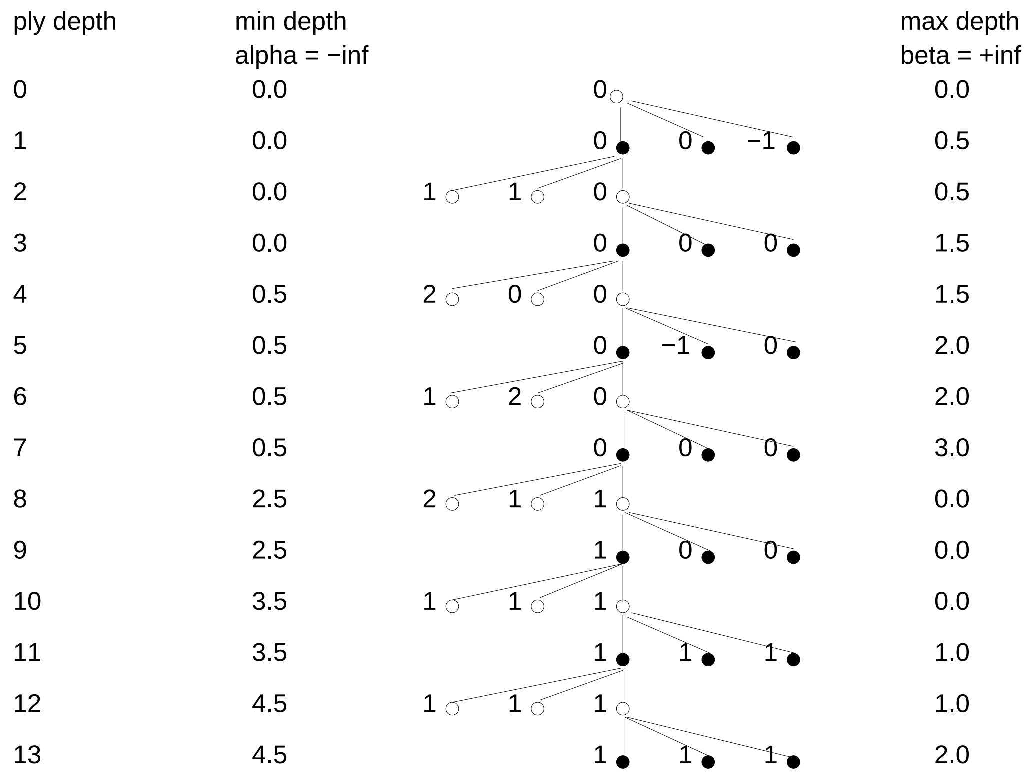


Figure 9: Discovering a combination

All of the examples shown in this section share a common framework. As before, open circles represent max nodes and closed circles represent min nodes. The root position in all figures is a max node. All the examples show the *static* value of each node just to the left of that node. Note, for example, that static value of the min node at the sixth ply in figure 9 is not the minimum of the static values of its children. All the examples show a single variation plus sibling nodes to the nodes on the variation. Together with the α and β values, also shown in the figure, this is all the information used by the ABC procedure to determine depth. The ABC procedure used space that is linear in the classical ply depth of the search. The nodes in the diagram are classified in the obvious way into backbone nodes and sibling nodes. Whether or not a sibling is an "option" depends on the value of that sibling and the current estimate of the value of the root node. Let n be a variation node whose depth is being measured. In the case where $\alpha < s_n < \beta$ the root estimate is s_n, the static value of n. If $s_n < \alpha$ then the root estimate is α and if $\beta < s_n$ then the root estimate is β. If $s_n \leq \alpha$ then the max depth is not measured and if $\beta \leq s_n$ then the min depth is not measured. Each figure shows a value for α and a value for β to be used at all nodes in the variation. In practice the α and β values can change during the search. For the sake of simplicity we assume they are given at the top level as an a-priori window and that they do not change as during the exploration of the variation shown. A sibling that is a child of a max node is an option if it is no more than δ less than the root estimate. A sibling that is a child of min node is an option if it is no more than δ greater than the root estimate. We assume that the singular margin δ is $\frac{1}{2}$ so if the static values are integers then any sibling with static value less than the root estimate is can not be an option and any sibling with static value greater than the root estimate can not be a min option. For backbone node the figures show both the min depth and max depth of that node as measured by the ABC procedure. The letter "N" is used to indicate that the depth is not measured. Recall that max depth is the sum over all max nodes above the one being measured of $S(k)$ where k is the number of options at the parent. The depth numbers are computed assuming that $S(k) = \frac{k}{2}$ where k is the number of options so that each option increases the depth by $\frac{1}{2}$. The search terminates when both the max and min depth reaches 2.0. With a branching factor of 3 max depth can increase by at most 1.0 at each max ply and min depth can increase by at most 1.0 at each min ply. So requiring a depth of 2.0 for both max and min corresponds to a classical depth of 4 ply. Figure 8 shows the "nominal" case of a quiescent position. Both the min and max depths reach 2.0 at the fourth ply.

Now consider the example shown in figure 9. Note that early in the search the static values of nodes on the variation are 0. Assuming that the root value is 0, the max player has many options. So the max depth increases rapidly, reaching 2.0 at the fifth ply and 3.0 at the seventh ply. However, the min player is fighting for his life — there are very few options to the variation shown. The min depth does not climb above 0.5 until the eighth ply. At the eighth ply the static value of the nodes on the variation shift from 0 to 1. This means that the estimate of the root value shifts from 0 to 1. The min player, who was fighting for his life, has lost this critical variation. To establish that the root value is 1 we must establish that the max player can "hang on" to this advantage. Suddenly the max player has very few options. In order to hang on to the value 1 the max player is essentially forced down this variation. So at the eighth ply the max depth immediately goes to 0.0. The min player on the other hand suddenly has lots of options. If the root value is 1 then the min player is free to select moves not on this variation which also have value 1. At the eighth ply the min depth jumps to 2.5. From the root to the seventh ply the search is continued because of insufficient min depth — the min player does not have an adequate strategy for defending a root value of 0. At the eighth ply we have sufficient min depth — the min player has a more than adequate strategy for defending a value of 1. However, at the eighth ply the max depth, which was adequate up to this point, is suddenly insufficient — although the max player has an adequate strategy for defending a value of 0 the max player does not have an adequate strategy for defending a value of 1. An adequate max strategy for defending the value 1 is not achieved until the thirteenth ply.

Figure 9 shows a 13 ply search that is required to establish a conspiracy depth of 2.0. This might seem disturbing — ABC seems in danger of generating huge search trees to achieve even small conspiracy depths. We believe that this will not be a problem in practice. Under optimal move ordering the first variation considered will be the principle variation and the static value encounter at the end of that variation will be the true root value. Of course move ordering is not perfect in practice, but hash tables and iterative deepening can be used to make move ordering nearly optimal. The search will tend to go deepest down the principle variation because static values on the principle variation are near the true root value and the principle variation involves making the best choice at each move so options tend to be limited. At moves off the principle variation, and under optimal move ordering, static values tend to be either significantly less than α or significantly larger than β. If the values are greater than β then the search terminates as soon as the max depth reaches a sufficient value (ignoring max depth). But if the values are greater than β then β is used as the estimate of the root value and the max player has many options. Hence in a region of the tree where the static values are greater than β max depth increases rapidly and the search terminates. A similar statement holds for regions of the tree with static value less than α.

Figure 10 shows a search that is similar that that shown in figure 9 except that the move considered from the root node is a "sacrifice". It has a static

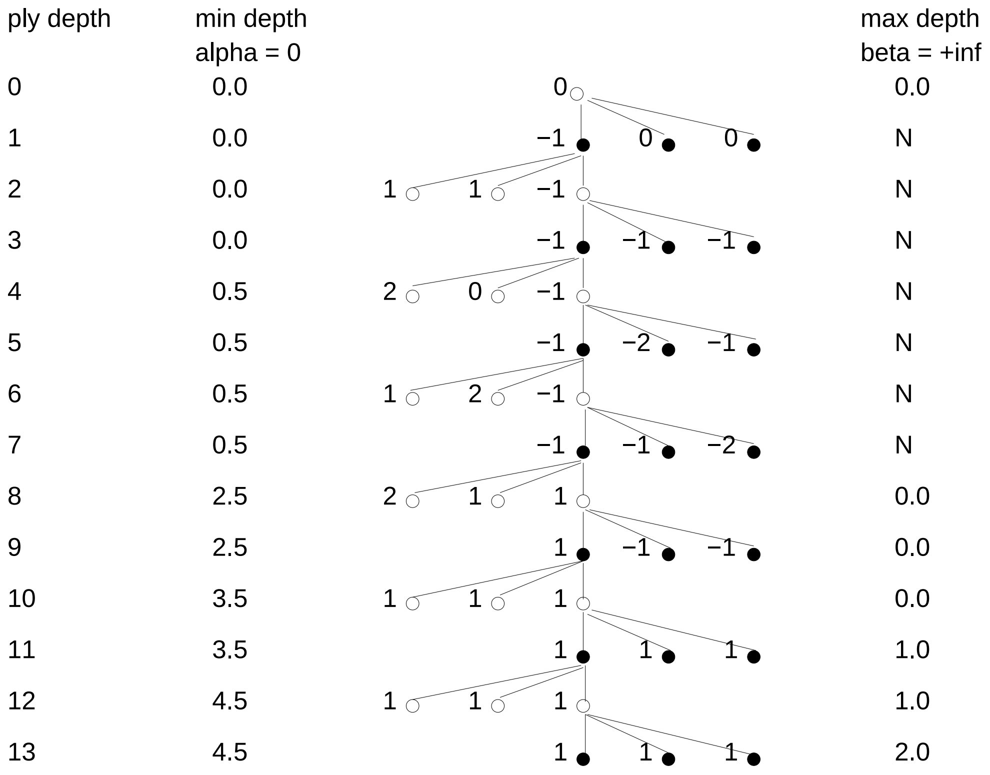


Figure 10: A successful sacrifice

value less than the static value of other options at the max node. This example is intended to show how the ABC procedure can search sacrificial moves deeply under appropriate circumstances. In figure 10 the max player selects a move with value -1 when moves with value 0 are available. Furthermore, seeing the true value of this sacrificial move requires an 8 ply search. The search is being done to a depth of 2.0 for both min and max depth. This is nominal depth of only four ply so it may seem surprising that a sacrifice move is carried deep enough (eight ply) to see its value. For the first seven ply the static values of the backbone positions are less than α which is 0. In this case the max depth is ignored and only min depth is computed. As the depth calculation at ply 4 shows, min depth is calculated relative to α — we are measuring the safety of the min strategy for establishing the value α. As the figure shows, however, the backbone nodes represent a critical line for the min player. Even though the static values on the backbone line are less than α, the min player has very few options if the max player takes this line of play. Because of the lack of min options, the min depth grows very slowly. The fact that options for the max player have low static values does not change the fact that if the max player decides to take this line of play the moves for the min player are largely forced. At the eighth ply the static value becomes 1 rather than -1. This means that this critical line of play for the min player has failed and it now seems quite possible that this is winning line for the max player. The situation is similar to that shown in figure 9. The max depth, which was not even measured in the first seven ply, suddenly becomes important at the eighth ply. At the eighth ply the estimate of the root value becomes 1 and we must continue the search until the max player has enough options for defending a value of 1. This happens at the thirteenth ply. Notice that only very special cases of sacrifices are searched deeply. The sacrifice must generate "pressure" on the opponent so that the safety of the opponent under the sacrifice is in question.

Figure 11 shows a case where we are given a narrow initial α-β window with $\alpha = 0$ and $\beta = 1$. For integer static values (and with $\delta = \frac{1}{2}$) The procedure must either find a min strategy for achieving 0 or a max strategy for achieving 1. At nodes where the static value is 0 or less, the max depth is not measured. At nodes where the static value is 1 or greater the min depth is not measured. In measuring the min depth at a node with static value 0 or less any sibling node which is a child of a min node and has static value 0 or greater is an option for the min player, regardless of the static value of the node being examined. Note that the min depth climbs rapidly at the end of the search. If $\alpha = -\infty$ then the min moves at the end of the search would be considered to be forced and the depth would not climb. The example shown in figure 11 also shows a case where an apparently successful move for the max player fails after a deep search. At ply one through seven the static value of the backbone node is 1. It appears that the max player has a winning strategy. However, there are very few max options for achieving the value 1 so the max depth grows slowly. This is a critical line if the value 1 is to be achieved. The line of attack fails at the eighth ply. From the eighth to the twelfth ply the static value of the backbone nodes are less than α. In this case the max depth is not measured and the min depth increases rapidly because there are several min options for achieving the value α at the tenth and twelfth ply.

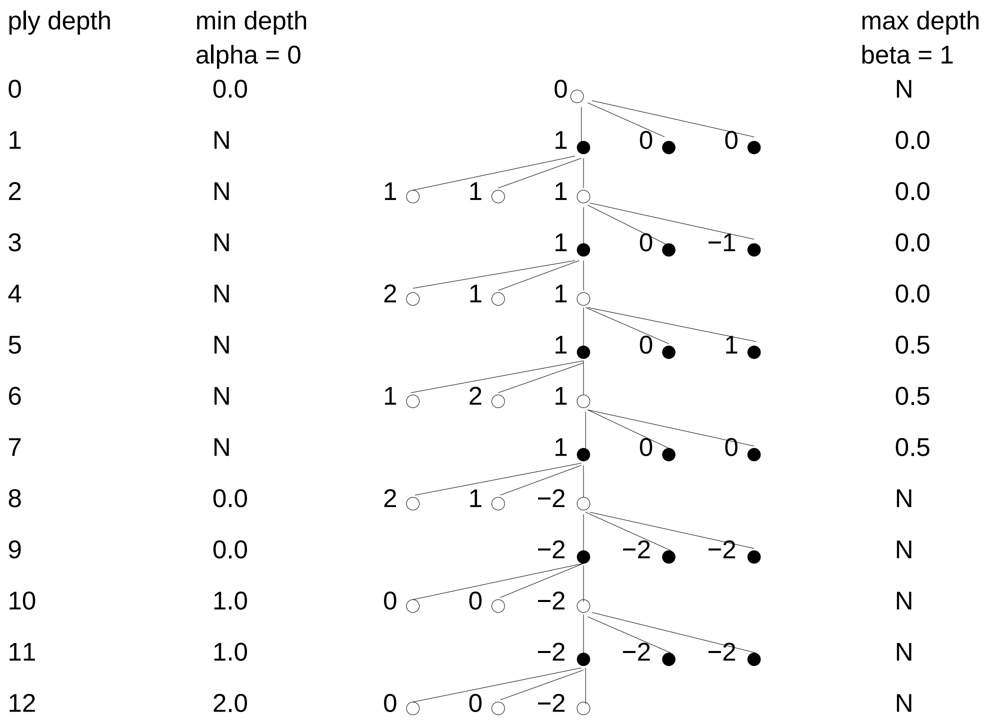


Figure 11: A failed attack within a narrow window

The last three examples show very deep searches generated by relatively modest depth inputs. It should be noted that these are exceptional cases near the principle variation of the play. As noted above, in quiet positions, or in regions of the tree with static values less than α or greater than β, depth should increase rapidly and the search should be shallow. Under intelligent move ordering only the lines that seem critical are explored deeply. None the less, we expect that in tactically complex positions very large searches will be generated with modest depth requirements.

9 Chess Heuristics for Variable Depth

In this section we consider classical heuristics for nonuniform tree growth. We consider four heuristics — quiescence search, capture extensions, check extensions, and singular extensions. These are usually presented as chess specific heuristics. In this section we argue that they are all special cases of the game independent "conspiracy depth principle" — search should be continued as long as conspiracy depth is low.

Chess heuristics for searching to variable depth can be roughly divided into two categories — selectivity heuristics and extension heuristics. Selectivity heuristics control the termination of the search — either causing a line of play to terminate unusually early or causing a line of ply to be taken unusually deep. Under this classification, quiescence search is a selectivity heuristic. Extension heuristics are used to discount certain moves as a ply of search. For example, almost all competitive chess programs do not count a response to check when computing the depth of a given node in the search. This causes the tree under a response to check to be searched one ply deeper than it would without this check extension heuristic. We believe that all successful selectivity and extension chess heuristics can be viewed as special cases of the conspiracy depth principle.

9.1 Quiescence Search and Capture Extensions

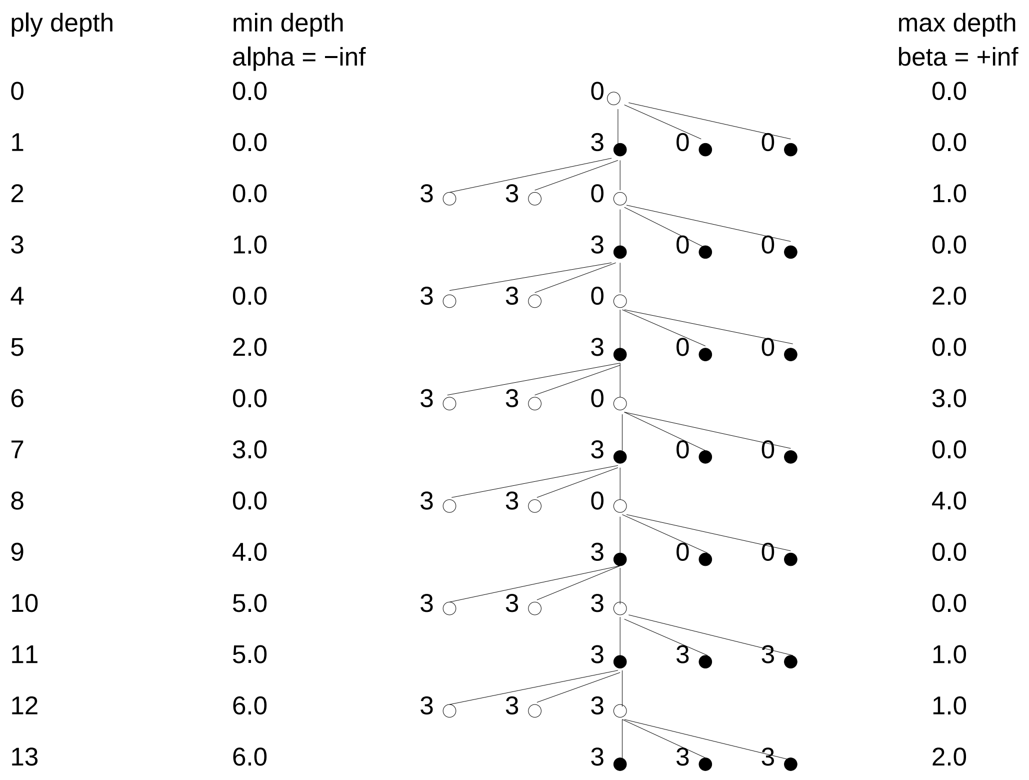
Shanon and Turing, in the earliest papers on computer chess, suggested that "forced" variations should be searched beyond the horizon, i.e., to a depth greater than the nominal ply depth of the search [Shanon, 1950], [Turing *et al.*, 1953]. The most immediately successful implementation of this idea has become known as quiescence search. As defined in section 8, the quiescence value of a position is the value computed from searching the stand-pat or capture tree from that position — at each position the player to move can either elect not to move (stand pat) or can elect a capture move. The stand-pat or capture tree can be searched quickly in practice, especially when static values fall outside the α-β window. Almost all competitive programs use some form of quiescence values at the leaves of the search. We show here that if the ABC procedure is used with static values (rather than quiescence values as recommended in practice) ABC search automatically simulates quiescence search.

The main observation is that capture moves tend not to increase conspiracy depth. This observation underlies both quiescence search and attempts to *capture extensions* — extensions granted at capture moves. Although quiescence search is almost universally used, generating an extension at every capture move tends to generate too large a search tree. Heuristics have been proposed for restricting capture extensions in some way to avoid the avalanche of extra positions [Kaindl, 1983]. Here we show that the ABC procedure will automatically search deeper under a fairly restricted class of capture moves.

Consider the situation shown in figure 12. This shows a very long capture sequence. In this sequence the nodes are labeled with classical static values rather than quiescence values. Every move on the backbone down to the ninth ply is a capture move. Since the ABC procedure measures depth on the principle variation relative to the static value of the node whose depth is being measured, and since the static value of the backbone nodes are oscillating between 0 and 3, the min depth and max depth also oscillate as the procedure descends the exchange sequence. However at no point are both depths at least 2.0 until the thirteenth ply. From the ninth ply forward the exchange sequence stops and the depths stabilize. At this point the exchange is resolved as a winning combination for the max player. Although the min depth is quite adequate — the min player can easily defend a value of 3 — the max depth at the ninth ply is zero. The search must be continued until an adequate number of options for the max player to defend the value 3 has been established. In the example shown the exchange sequence occurs at the root of the tree. However, a similar failure to increase depth occurs if the exchange sequence is near the leaves of the tree.

It is interesting to note that if the static value is greater than or equal to β then min depth is not measured. Note that in figure 12 the max depth tends to be high at max nodes (where the value is low). If, in an exchange sequence, there is a max node with value above β then the max depth will be high and the min depth will be irrelevant so the search is likely to terminate. This is exactly what happens in the interaction between the stand-pat option and the α-β window in computing quiescence values. If the static value at a max node is greater than or equal to β then the stand-pat option causes a cut off and no search need be done. Near the leaves of the tree quiescence search with the stand-pat option and α-β pruning efficiently simulates ABC.

If the ABC procedure uses quiescence values instead of static values then the values of the backbone nodes would very likely all be 3. However, the values of the sibling nodes (off the backbone) would very likely remain unchanged. In this case the min depth would increase steadily and the max depth would remain 0.0 until the ninth ply after which it would increase steadily to 2.0 at the thirteenth ply. This corresponds to a special case of



ply depth	min depth alpha = −inf	max depth beta = +inf
0	0.0	0.0
1	0.0	0.0
2	0.0	1.0
3	1.0	0.0
4	0.0	2.0
5	2.0	0.0
6	0.0	3.0
7	3.0	0.0
8	0.0	4.0
9	4.0	0.0
10	5.0	0.0
11	5.0	1.0
12	6.0	1.0
13	6.0	2.0

Figure 12: An exchange sequence starting at the root

classical capture extensions — the search is extended below capture moves. However, the circumstances under which ABC extends search under capture moves are fairly restrictive. The depth of the player making the capture move must be the controlling factor in determining the depth of the search. For example, if the capture move is a move by the max player then it must be the max depth that is determining the depth of search. Furthermore, the capture usually only causes an extension if it is a move from a position with static value less than $v_r - \delta$ where v_r is the estimate of the root value and δ is the singular margin.

9.2 Check Extensions

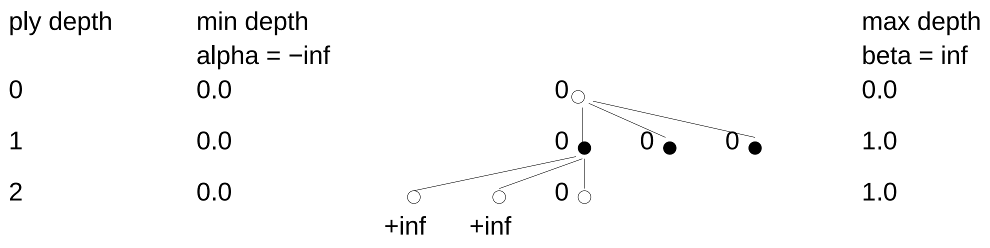


Figure 13: An aggressive check

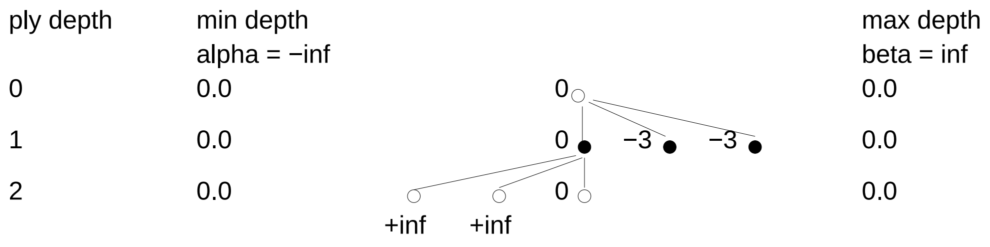


Figure 14: A defensive check

Conspiracy depth tends not to increase at moves which are responses to check. Figures 13 and 14 show two different ways that check moves can affect conspiracy depth. Figure 13 shows the effective use of a check as an aggressive move. The check is performed by the max player at the first ply. Since the min players moves are highly restricted at the second ply, min depth does not increase at the min players move. This will the search below the check move to go deeper in regions where search termination is controlled by min depth. In general, at a move out of check the depth of the player moving out of check does not increase significantly. Note that the use of two depths allows a more refined application of the check extension heuristic.

Figure 14 shows a defensive check move. The check is performed by the max player at the first ply under tactical pressure — all of the options to the checking move lose a piece. In this case both the checking move and the response to check fail to increase conspiracy depth So the search beneath the response to check move will be extended in both the regions where termination is controlled by min depth and in the regions where termination is controlled by max depth.

9.3 Singular Extensions

Conspiracy depth does not increase at forced moves. This is the fundamental principle of conspiracy depth and is also the fundamental principle behind singular extensions. The great success of the current world champion computer chess program, Deep Thought, is considered to be due in part to the use of *singular extensions* [Kopec, 1990], [Anantharaman *et al.*, 1990]. A max node is called *fail high singular* for a depth d search if it appears to be forced, i.e., for all siblings n' of n we have

$$v_{d-r}[n'] < v_r - \delta$$

where $v_{d-r}[n]$ is the min-max value of n searched to depth $d-r$, the value v_r is an estimate of the root value of the search and δ is a fixed number called the *fail high singular margin.* The dual definition would hold at min nodes. The reason for calling this "fail high" singular rather than simply singular is explained below. This definition is closely related to the definition of the option set $O[n,\ T]$ given in section 5. If we replace $v_{d-r}[c']$ by the static value (or quiescence value) $s_{n'}$, then we get that a move is singular if and only if the option set $O[n,\ T]$ is empty. Although conspiracy theory was originally developed independently of the singular extension heuristic, the term "singular margin" for the parameter δ is taken from the above definition by Anantharaman et. al.

One difference between singular extensions and conspiracy depth is that conspiracy depth, as formulated in this paper, is based on a static rather than dynamic computation of the option set — a static evaluator is used rather than a dynamic search to compute the values of the options. This allows depth to be computed more efficiently and simplifies the structure of the ABC procedure. However, at shallow nodes, where one can afford spending considerable time evaluating the option set, the dynamic approach is more accurate. It seems that some variant of the ABC procedure could be defined to use dynamic evaluation of the option set.

Another difference between conspiracy depth and singular extensions is the fact that singular extensions are based on a Boolean decision at a given move — it is either singular or not singular. Conspiracy depth is fundamentally based on "fractional extensions" — measuring depth in factions of a ply. Anantharaman et al. mention the use fractional extensions as a possible enhancement of singular extensions.

Of course the most significant difference between singular extensions and conspiracy depth is that conspiracy depth involves two depths rather than one. Under a two depth approach to singular extensions a min depth extension would be granted at singular min moves and a max depth extension would be granted at singular max moves.

The one depth formulation of singular extensions used by Anantharaman et al. causes some difficulties. To simplify the discussion we assume that the search is optimally ordered. Near optimal ordering can be achieved in practice. An optimally ordered search tree can be divided into three regions — the principle variations where computed values equal the root value, the non-PV max strategy where β equals the root value and all computed values are at least β, and the non-PV min strategy where α equals the root value and all computed values are no larger than α. On the principle variation the search termination test involves both the min depth and the max depth. In the non-PV max strategy successful termination tests involve only the max depth. In the non-PV min strategy successful termination tests involve only the min depth.

Consider the non-PV max strategy. Since termination in the max strategy is determined by max depth, and since max depth only increases at moves for the max player, it seems clear that for non-PV max strategy nodes singular extensions should only be granted for forced max moves. Similarly, in non-PV min strategy nodes singular extensions should only be granted for forced min moves. This is in fact what is done by Anantharaman et al. A max node n where the search value $v_n \geq \beta$ is called a *fail-high* node. In an optimally ordered search, the fail-high max nodes are exactly those max nodes in the non-PV max strategy. The term "fail-high" comes from the negamax formulation of α-β search. Under the negamax formulation of α-β a min node n where $v_n \leq \alpha$ is also called a fail-high node. In an optimally ordered search the fail-high nodes are exactly the non-PV nodes which are either max nodes in the max strategy or min nodes in the min strategy. For non-PV nodes Anantharaman et al. only grant singular extensions at the fail-high nodes. This is exactly as prescribed by the conspiracy depth analysis. However, Anantharaman et al. do not mention any theoretical justification for the restriction of singular extensions to fail-high nodes and even suggest that some formulation of singular extensions for the fail-low nodes can be found. For nodes on the PV it seems very likely that Anantharaman et al.'s use of singular extensions is hampered by the lack of two depth measurements.

The fact that two depths are needed on the PV seems to be reflected in their algorithm by the use of a different singular margin for the PV nodes.

10 Efficiency Considerations and Static Option Estimators

In the ABC procedure calculations of conspiracy depth control the shape of the search tree. An exact calculation of conspiracy depth requires the calculation of static values of option moves. In most cases these are moves not examined by classical α-β (the option nodes are not part of the min or max strategy). Clearly there is a tradeoff between the cost of refining the shape of the search and the improved performance gained by such refinement. Since search speed is very important in computer chess, it seems important to consider techniques for reducing the overhead of the depth calculations.

10.1 Incremental Depth Calculations

Under nearly optimal move ordering, or under narrow window searches, the vast majority of depth computations will be performed relative to the values α and β. The procedure can be easily modified to take two additional parameters d_α and d_β. In any call to ABC these parameters should have the following values.

$$d_\alpha = \text{min-depth}(\alpha,\ \text{min-options})$$

$$d_\beta = \text{max-depth}(\beta,\ \text{max-options})$$

Clearly the numbers d_α and d_β could be computed from the other parameters of the procedure. However, passing them explicitly greatly improves the efficiency of cases 2 and 3 of the termination test. Under good move ordering, or with narrow window searches, these are the cases that will be used at the vast majority of nodes. Given the numbers d_α and d_β the termination test will almost always be done using two simple comparisons. Furthermore, in the case where the α-β window for the call to a child node is the same as the α-β window for the parent node (again the vast majority of nodes) the parameters d_α and d_β can be updated incrementally.

10.2 Quiescence Values

The ABC procedure should use quiescence values, as defined at the beginning of section 8, rather than classical static values. Quiescence search gives an efficient approximation of ABC search to shallow depths. Using quiescence values rather than static values should greatly improve the performance of the procedure by eliminating depth calculations during quiescence search and by improving the estimation of the value of option nodes. Unfortunately, the use of quiescence values for option nodes also increases the overhead of the depth calculations. A static option estimator, as discussed below, can be used to reduce the cost of measuring option values at nodes near the leaves of the tree.

10.3 Using a Static Option Estimator

The most significant source of overhead in the depth calculations is the computation of the values of option nodes. We believe that this overhead can be greatly reduced with the use of a *static option estimator*. A static option estimator is function O_s such that for any node n, $O_s[n]$ is a list of values representing an estimate of the values of the children of the node n. We would suggest computing $O_s[n]$ in a way that combines check extensions with the null move heuristic. The null move value of a max node n is defined here to be the quiescence value of the node n assuming the min player is allowed to move first from that position. The null move value of a min node is defined in the dual way.

- If the player to move is in check then $O_s[n]$ is two copies of the quiescence value of n.
- If the player to move is not in check then $O_s[n]$ is three copies of the quiescent value of n plus thirty copies of the null move value of n.

For example, consider a max position in which one of the max pieces is in danger of being captured. In this case the null move value of the position is considerably less than the quiescence value of the position. Most max moves will lose the piece and yield roughly the same value as the null move value of the max position. The children of the max node should have roughly the value distribution indicated above. If the null move value is above β at a max node then max depth will increase significantly even though a max piece is danger of being captured.

Intuitively, the nodes of search tree can be classified into three types: leaf nodes, near leaf nodes, and shallow nodes. The near leaf nodes are those nodes at which the min and max depth are nearly sufficient to terminate the search. The shallow nodes are all the nodes other than leaves and near leaves. There are vastly more leaf and near leaf nodes than shallow nodes. The cost of evaluating option nodes at shallow nodes is negligible compared to the time spent evaluating leaf and near leaf nodes. Therefore the ABC procedure can use quiescence values in counting options at shallow nodes without incurring undue overhead. At near leaf nodes the cost of evaluating option nodes can become significant. At near leaf nodes it seems best to use the static option estimator. When computing the value of a near leaf node n one can compute $O_s[n]$ once and use this value list as the option list for all of the children of n.

11 Empirical Results

12 Summary

The idea that all lines of play should be searched to a uniform depth is ludicrous to most chess players. The idea that forced moves should be searched more deeply that unforced moves goes back to the original papers on computer chess by Shanon and Turing [Shanon, 1950], [Turing *et al.*, 1953]. This basic idea underlies most of the heuristics for search to variable depth that have been used in competitive programs, quiescence search, check extensions, and singular extensions being the most notable examples. The main innovation of the ABC procedure is the introduction of two separate measures of depth and the realization that the forcedness of a given move influences only one of these two depths. Conspiracy theory, as the theoretical underpinning of the ABC procedure, also gives considerable guidance in the construction of heuristic depth measures.

The task of α-β search is to find a strategy for the max player and a strategy for the min player that together establish the value of a given position. The max strategy provides a lower bound and the min strategy provides an upper bound. The max depth of a max strategy provides a measure the safety of that strategy — the deeper the strategy the less likely it is to fail. Similarly, the min depth of a min strategy is a measure of the safety of that strategy. The basic observation behind ABC search is that if a max move in a max strategy is forced then the strategy is "frail" — it is more likely to fail than if the max moves are not forced. A forced min move in a max strategy does not influence the safety of the strategy — if anything it makes the strategy safer. Hence max strategies in which the max moves are forced should be searched deeply independent of the forcedness of min moves.

We are quite confident in the two depth approach to α-β search. We are far less confident in the details of the ABC procedure specified in this paper. For example, it may be better to use a more dynamic approach to measuring forcedness as is done with singular extensions. There are almost certainly better static option estimators than the one described here. We look forward to the evolutionary development of variants of the ABC procedure.

References

[Anantharaman *et al.*, 1990] T. S. Anantharaman, M. S. Cambell, and F h. Hsu. Singular extensions: Adding selectivity to brute force search. *Artificial Intelligence*, 43(1):99–110, 1990.

[Baum and Smith, 1993] E. B. Baum and W. D. Smith. Best play for imperfect players and game tree search. submitted for publication, available for anonymous ftp from external.nj.nec.com as pub/eric/papers/game.ps.Z, 1993.

[Kaindl, 1983] Herman Kaindl. Searching to variable depth in computer chess. In *IJCAI-83*, pages 760–762, 1983.

[Kopec, 1990] D. Kopec. Advances in man-machine play. In T. Anthony Marsland and Jonathan Schaeffer, editors, *Computers, Chess, and Cognition*, pages 10–32. Springer-Verlag, 1990.

[McAllester, 1988] David A. McAllester. Conspiracy numbers for min-max search. *Artificial Intelligence*, 35(3):287–310, July 1988.

[Schaeffer, 1990] Jonathan Schaeffer. Conspiracy numbers. *Artificial Intelligence*, 43(1):67–84, 1990.

[Shanon, 1950] C. E. Shanon. Programming a computer to play chess. *Philosophical Magazine*, 41(7):256–275, 1950.

[Turing *et al.*, 1953] A. M. Turing, C. Strachey, M. A. Bates, and B. V. Bowden. Digital computers applied to games. In B. V. Bowden, editor, *Faster than Thought*, pages 286–310. Pitnam, 1953.

Appendix: Proofs of the ABC Search Theorems

First we restate the two theorems regarding the ABC procedure along with the definitions involved in the statements of the theorems.

> **Definition:** A tree T is said to *contain* the max strategy $T_{\max}$ if the two trees have the same root and every leaf node of $T_{\max}$ is a leaf node of T. A similar definition applies for min strategies.
>
> **Definition:** A tree T establishes a value v up to depth $<d_{\max},\ d_{\min}>$ if T contains a max strategy $T_{\max}$ and min strategy $T_{\min}$ both of which have value v and such that $D_{\max}[T_{\max}] \geq d_{\max}$ and $D_{\min}[T_{\min}] \geq d_{\min}$.
>
> **Theorem:** If the top level call
>
> $$\text{ABC}(\text{root},\ -\infty,\ +\infty,\ d_{\max},\ d_{\min},\ \textbf{nil},\ \textbf{nil})$$
>
> returns value v then the tree of nodes examined by the search establishes v up to depth $<d_{\max},\ d_{\min}>$.
>
> **Definition:** A search is *optimally ordered* if at each internal node n we have that v_n equals v_c where c is the first child of n examined by the procedure.
>
> **Theorem:** Let T be the tree of nodes examined by a search using depth parameters $<d_{\max},\ d_{\min}>$. If the search is optimally ordered then no proper subtree of T establishes a value up to depth $<d_{\max},\ d_{\min}>$.

To prove these theorems we need some additional terminology. Let n be any node examined during the search. Let α_n and β_n be the α and β parameters passed to the ABC procedure in the evaluation of n. let s_n be the static value of n and let v_n be the value returned by the call to ABC on n. Note that values less than α_n are replaced by α_n and values greater than β_n are replaced by β_n. We therefore have $\alpha_n \leq v_n \leq \beta_n$. Let v_r be the value returned by the search.

To prove the first theorem consider an arbitrary application of ABC. Let $T_{\max}$ be the subset of the tree defined by starting at the root and taking the child with largest min-max value at each max node and all children at each min node. If a max node has two children that are tied for having the largest min-max value then $T_{\max}$ includes that child which was examined first by the procedure. The tree $T_{\min}$ is defined in the dual way. We will show that $T_{\max}$ and $T_{\min}$ are the desired strategies. First, it should be clear that both $T_{\max}$ and $T_{\min}$ are max and min strategies respectively and that both have value v_r. It remains only to show that $D_{\max}[T_{\max}] \geq d_{\max}$ and $D_{\min}[T_{\min}] \geq d_{\min}$. We consider only $T_{\max}$, the proof for $T_{\min}$ is similar. We make the following claims for any node n in $T_{\max}$.

$$\alpha_n < v_n$$

$$v_r \leq v_n$$

$$v_r \leq \beta_n$$

The first condition follows from the fact that if $v_n \leq \alpha_n$ then n will be pruned from the max strategy. The second condition follows from the fact that in a max strategy the max player only has one move at any given position so the value of a max strategy is computed to be the minimum of the value of all the nodes in the max strategy. The third condition is a little trickier. If $\beta_n = +\infty$ then the result is trivial. If $\beta_n \leq +\infty$ then there exists some min node m above n such that $v_m \leq \beta_n$. But in this case v_m must be a member of the max strategy $T_{\max}$ and we have $v_r \leq v_m$ so $v_r \leq \beta_n$. We must show that for every leaf node n in $T_{\max}$, $D_{\max}[n,\ T_{\max}] \geq d_{\max}$. Since n is a leaf node it must have passed the termination test at step 1 of the ABC procedure and $v_n = min(\beta_n,\ s_n)$ where s_n is defined above as static value of n. Since $v_n > \alpha_n$ it must have passed either step 2 or step 4 of the termination test. If it passed step 2 then the depth of the node in $T_{\max}$ was measured relative to the value β_n. Since $v_r \leq \beta_n$ the depth measured relative to v_r, which is the true depth, must be at least as large as the depth measured relative to β_n. If node n passed step 4 of the termination test then the depth of n in $T_{\max}$ was measured relative to v_n. Since $v_n \leq v_r$ we again get that $D_{\max}[n,\ T_{\max}] \geq d_{\max}$. Since $v_n \leq v_r$ we again have that $D_{\max}[n,\ T] \geq d_{\max}$. Note that the depth always measured relative to the minimum of β_n and v_n and hence the procedure always uses the tightest possible upper bound on v_r.

We now prove the second theorem. As in classical α-β search, it is easy to show that under optimal ordering the tree T of all nodes examined is the union of a max strategy $T_{\max}$ and a min strategy $T_{\min}$. By the proof of the first theorem, both these strategies have value v_r and $D_{\max}[T_{\max}] \geq d_{\max}$ and $D_{\min}[T_{\min}] \geq d_{\min}$. Now let T' be a proper subtree of T and assume that T' establishes a value for the root. In order for T' to establish a value it must contain both a max strategy and a min strategy. So T' can be written as $T'_{\max} \cup T'_{\min}$ where $T_{\max}$ and $T_{\min}$ are max and min strategies that are subtrees of $T_{\max}$ and $T_{\min}$ respectively. We now consider two cases. First, we suppose that the min-max value of T' is different from the min-max value of T. In this case we consider the *principle variation*, i.e., the intersection of $T_{\max}$ and $T_{\min}$. The *principle leaf* of T is defined to be the leaf node that is a member of principle variation. The principle leaf of T' is defined to be the element of the principle variation that is a leaf node in T'. Since T' establishes a value, the value of the $T'_{\max}$ and $T'_{\min}$ must be the same. This implies that the min-max values of T and T' must both be equal to the static value of their respective principle leaf nodes. So in the case where the values of the two trees are different we must have that the principle leaf of T' is a proper ancestor of the principle leaf of T. Let n be the principle leaf of T'. In an optimally ordered search the termination test measures the depth of every node on the principle variation relative to the assumption that the root value is the static value of that node. Since the termination test failed on node n we have that either $D_{\max}[n, T'_{\max}] < d_{\max}$ or $D_{\min}[n,\ T'_{\min}] < d_{\min}$. Now we consider the case where the min-max value of T' is the same as the min-max value of T. In this case let n be a leaf node in T' that is an internal node of T. The existence of such a node is guaranteed by the statement that T' is a proper subtree of T and that both T and T' are unions of a max and min strategy. If n is a member of the principle variation then by the preceding argument either $D_{\max}[n,\ T'_{\max}] < d_{\max}$ or $D_{\min}[n,\ T'_{\min}] < d_{\min}$. So we can assume that n is not a member of the principle variation. We consider only the case where n is a leaf node of $T'_{\max}$, the case where n is a leaf of $T'_{\min}$ is similar. Since the value of T' equals the value of T equals v_r, the value of $T'_{\max}$ must equal v_r. Since the value of a max strategy is the minimum of the values of the leaves, we must have $v_r \leq s_n$. Furthermore, in a perfectly ordered search we have $\beta_n = v_r$ for every node n in $T_{\max}$ not in the principle variation. Putting these two facts together we get that $\beta_n = v_r \leq s_n$. Also, for a perfectly ordered tree we have that $\alpha_n = -\infty$ for every node n in $T_{\max}$. This implies that the termination test on node n was based on step 2 of the termination test procedure and therefore measured only the the max strategy depth and measured that depth relative to β_n which equals v_r. Since the termination test failed, we have $D_{\max}[n,\ T_{\max}] < d_{\max}$.