Artificial Intelligence Project--RLE and MIT Computation Center

Memo 30 (Revised)--

THE α-β HEURISTIC

by D. J. Edwards and T. P. Hart

October 28, 1963

(This memo is a revised version of The Tree Prune (TP) Algorithm, AI Memo 30 December 4, 1961.)

The α-β Heuristic

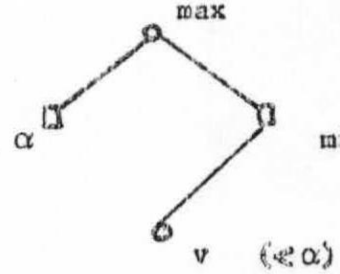
by D. J. Edwards and T. P. Hart

The α-β heuristic is a method for pruning unneeded branches from the move tree of a game. The algorithm makes use of information gained about part of the tree to reject those branches which will not affect the principle variation.

The reasoning behind the α-β heuristic is as follows:

a) if the maximizing-player finds a move whose value is greater than or equal to the value of an alternate minimizing-player move found higher in the tree, he should not look further because the min-player would certainly take that alternate move.

b) if the min-player finds a move whose value is less than or equal to the value of an alternate max-player move found higher in the tree, he should not look further because the max-player would certainly take that alternate move.



Case (b) may be illustrated by the above piece of tree. The max-player has investigated the branch with value α and has looked at an alternate move from the point at which this branch was found. At the time the value v is established the maximizing player (whose point of view we are taking) sees that if he takes the right branch from the top node he is giving the other player the opportunity to take the v branch. But this will result in his getting v or less, and since he can get α by taking the left branch, he instantly decides that he won't make the move to the right under any circumstance.

The following definitions express the α-β heuristic:

```
vmax[pos;α;β] = [if final[pos;α;β] then evaluate[pos]
                 else vlmax[succ[pos];α;β]]

vlmax[lis;α;β] = [if null[lis] then α
                  else if vmin[car[lis];α;β] ≥ β then β
                  else vlmax[cdr[lis];
                             max[vmin[car[lis];α;β];α];
                             β]]

vmin[pos;α;β] = [if final[pos;α;β] then evaluate[pos]
                 else vlmin[succ[pos];α;β]]

vlmin[lis;α;β] = [if null[lis] then β
                  else if vmax[car[lis];α;β] ≤ α then α
                  else vlmin[cdr[lis];
                             α;
                             min[vmax[car[lis];α;β];β]]]
```

where:

pos stands for some representation of the current position.

lis is a list of position representations.

succ[pos] is a function which produces a list of positions which can be reached in one move from pos.

final[pos;α;β] is a predicate which may decide that the search is not to continue, e.g., when the end of the game has been reached.

evaulate[pos] provides some measure of goodness of the position (always from the point of view of maximizing player).

α and β are initially set at $-\infty$ and $+\infty$ respectively

This heuristic may readily be compared with a minimax search by observing that if the second clauses in the conditional expression for <u>vlmax</u> and <u>vlmin</u> are eliminated the resulting function is just minimax.

It is important to note that the ordering of the list of moves generated by <u>succ</u> is very important. The most gain is achieved from the heuristic when the right move is at the beginning of the list.

It turns out that at best, that is, in the case of perfect ordering the α-β heuristic can cut a tree's exponential growth rate in half, thus allowing almost twice the search depth for the same effort. More precisely, we have the following theorem.

<u>Theorem</u> (Levin): Let n be the number of plies in a tree, and let b be the number of branches at every branch point. Then the number of terminal points on the tree is

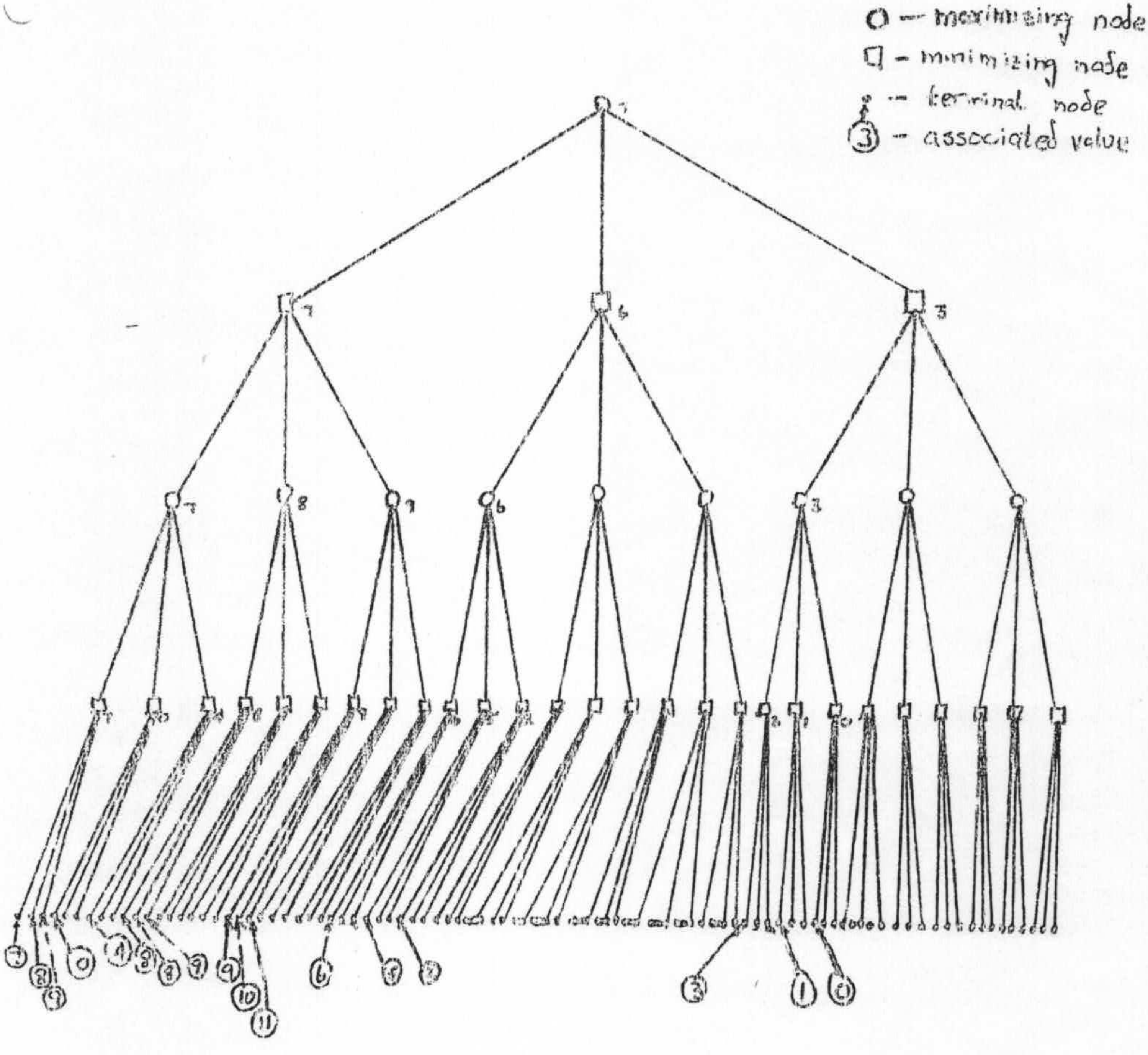
$$T = b^n$$

However, if the best possible advantage is taken of the α-β heuristic then the number of terminal points that need be examined is

$$T = b^{\frac{n+1}{2}} + b^{\frac{n-1}{2}} - 1 \qquad \text{for odd } n$$

$$T = 2b^{\frac{n}{2}} - 1 \qquad \text{for even } n$$

EXAMPLE



This example shows an ordered ternary tree where only the labelled terminal nodes need be examined. According to the formula for n even where b = 3 and n = 4 only 17 out of the 81 possible nodes need be examined and this is shown to be true in this case. Note that the unlabelled nodes may have any possible value and the result will still be unchanged.