32002: AI (Alpha-Beta Search) **Spring 2014**

Lecture 35: April 10, 2014

Instructor: K.R. Chowdhary *: Professor of CS*

35.1 Alpha-Beta Procedure

Alpha-Beta is a tree-search procedure that is faster than minimax but still equivalent in the sense that both procedures will always choose the same depth successor at best, and will assign the same value to it. Alpha-Beta is typically several orders of magnitude faster than minimax. It saves time by not searching certain branches of the tree. Under certain conditions the values of certain branches do not affect the value which is ultimately backed up to higher levels of the tree. Hence, there is no point in evaluating these branches. When the Alpha-Beta program detects these conditions, it stops work on one branch and skips to another. This event is called an alpha or beta cutoff.

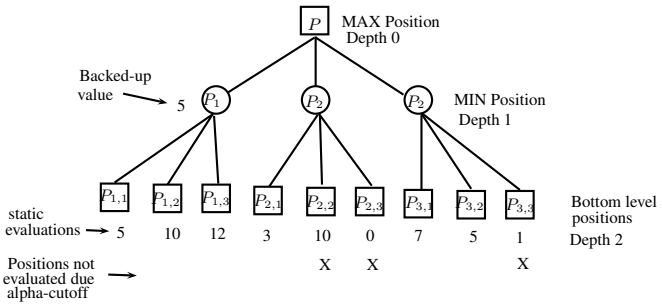

Figure 35.1: Alpha-Beta search.

At first it appears that *static evaluator* must be used on each leaf-node at the bottom of the search-tree. But, it is not required. To see how alpha-beta search works, consider the example shown in figure 35.1. Alpha-beta starts just like minimax procedure by evaluating all the successor of P_1. The minimum of these static vales is then backed-up to P_1, since P_1 is a MIN position. The backed-up value of P_1 is *alpha* and has a value 5 in this example.

Alpha is lower limit for the backed-up value of the top position, P. Since, P is max position, we back-up the value of the largest values successor of P. Since, we have evaluated only one successor at this time, we do not know that it will be 5 or larger. The value of alpha may change as the other successors are evaluated, but it can only increase, not decrease.

Having evaluated P_1, the procedure begin work on P_2. An *alpha cut-off* takes place at $P_{2,1} = 3$, as it is less than alpha. Since P_2 is a Min position, $V_{2,1}$ is an upper limit for V_2 (value of node P_2). Since V_2 is less than alpha, P_2 is definitely eliminated as a candidate for the largest valued successor of P. There is no point in evaluating the other successors of P_2 so the procedure begins work on P_3 next.

The above cut-offs saves the machine a good deal of time. The alpha cut-off at $P_{2,1}$ means that the machine

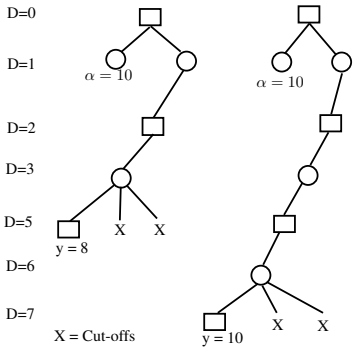

Figure 35.2: Deep Alpha cut-offs.

need not bother to evaluate $P_{2,2}$ and $P_{2,3}$. A second alpha cut-off occurs at $P_{3,2}$, which eliminates $P_{3,3}$. Thus, in the figure 35.1 alpha-beta search program would evaluate only six of the bottom level successors, while a minimax program would evaluate all nine.

Although the example is given only for a tree of three levels, it is clear that procedure will work just the same below any Max position P_x, at any depth in a large tree, provided only that there are at least two levels below P_x. If there were more levels below $P_{1,1}$ in the example, then we could use the back-up value of $P_{1,1}$ instead of static value. If there were more levels above P, then we could backup of P.

Moreover, it is possible to pass a value of alpha down from the top of a large tree. Thus, an alpha established at depth 1 could be used to produce cut-offs at depths 2, 4, and 6. These deep cut-offs are illustrated in figure 35.2.

Alpha is defined by the values of the successors of a Max positions (i.e., odd depths, here it is P_1, P_2, P_3 nodes), while alpha cut-off occur among the successors of a Min position (even depths, here it is $P_{1,1} \ldots P_{3,3}$) It is possible to define another variable, *Beta*, which is established at even depths and generates cut-offs at off depths. The action of beta cut-off is exactly inverse of alpha cut-offs.

The effect of alpha-beta cut-off is to make the tree space grow slowly with depth. Thus, the advantage over simple MINIMAX. It is about twice as good as at Maximum depth Dmax = 3 and about thirty times as good at Dmax = 6. This is typical. It is good to have depth dependent program. Such a measure we call as depth ration DR,

$$DR = \frac{\log N}{\log N_{MM}} \tag{35.1}$$

where, N is number of nodes at the bottom of tree, and N_{MM} is number of nodes at the bottom of tree in minimax search. $DR = [0, 1]$, is effective depth in comparison to minimux procedure.

Alpha-beta search is equivalent to minimax in the sense the two procedure will choose the same depth successor as best and will always give the same value for the successor.

35.1.1 Complexities of Alpbha-Beta

The benefit of alpha-beta pruning lies in the fact that branches of search-tree can be eliminated so that search-tree can be limited to more promising subtrees and deeper search can be performed in lesser time. Thus it belong to the *branch-and-bound* class of algorithms.

If move ordering for alpha-beta is optimal (best moves are always selected first), then the total number of leaf nodes to be evaluated are $O(b^{\frac{d}{2}}) = O(\sqrt{b^d})$. The explanation of above is that all the first players' moves must be studied to find the best one, but for each only the best second player's move is needed to refute all but the first (and best) player move-alphabet ensures that no second player moves need to be considered. Thus, the complexity $(Ob^{\frac{d}{2}})$ is possible if nodes are ordered. If they are at random then it is $O(b^{\frac{3d}{4}})$.

Thus, it can be demonstrated that the number of static evaluations needed to discover the best move in an optimally arranged tree is $O(b^{d/2})$. Let the static evaluator function is represented by a variable c, which needs to be maximized and minimized at different levels. We note in figure 35.1 that evaluation of c is not required at level 1 (shown with values 5 and ≤ 3 for minimizing level). This is because we do not need to evaluate for ≤ 3, as the value 5 is carried forward to the root. Since static evaluations are required only at the alternate levels, the depth of search reduces to half. Therefore the time *time complexity* is $O(b^{d/2})$.

35.1.2 Improving the efficiency of Alpha-Beta

The number of cut-offs generated by alpha-beta procedure depends on the order in which successors are evaluated. In figure 35.1, if machine had evaluated P_2, P_3 then P_1, alpha would have been 0, then 1, then 5, and there would have been no alpha-cut off.

This fact suggests the possibility of improving the alpha-beta procedure by ordering successors of a position in order to generate a large number of alpha-beta cut-offs.

This can, in fact, be done by ordering successors of a position in order to generate a large number of alpha and beta cut-offs.

The ordering can be achieved by ordering the nodes of search tree at their static values. The largest value successor of a max position is put first and reverse order is used for the successors of min position. this procedure is based on the assumption that their static value of a position is positively correlated with the deeper, backup value at that position. Since ordering obtained may not be correct at other levels, dynamic ordering may be adopted.

References

[1] Chowdhary K.R. (2020) Logic and Reasoning Patterns. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_11