Artificial Intelligence

Unit-III

Chap-II

Adversarial Search

Alpha–beta Pruning

- In case of standard ALPHA–BETA PRUNING minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.

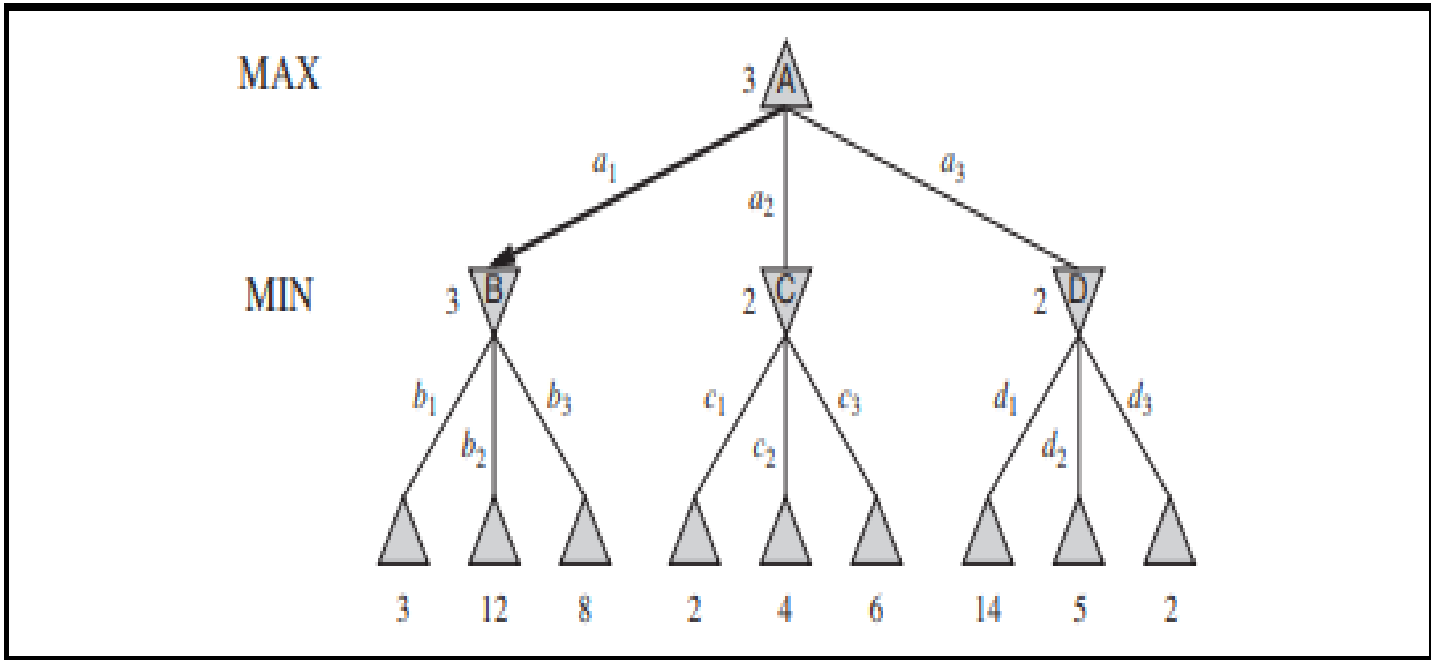

Figure A two-ply game tree. The △ nodes are "MAX nodes," in which it is MAX's turn to move, and the ▽ nodes are "MIN nodes." The terminal nodes show the utility values for MAX; the other nodes are labeled with their minimax values. MAX's best move at the root is a_1, because it leads to the state with the highest minimax value, and MIN's best reply is b_1, because it leads to the state with the lowest minimax value.

Alpha–beta Pruning

- Consider again the two-ply game tree from Figure Let’s go through the calculation of the optimal decision once more, this time paying careful attention to what we know at each point in the process.
- The steps are explained in Figure (P.T.O.).
- The outcome is that we can identify the minimax decision without ever evaluating two of the leaf nodes

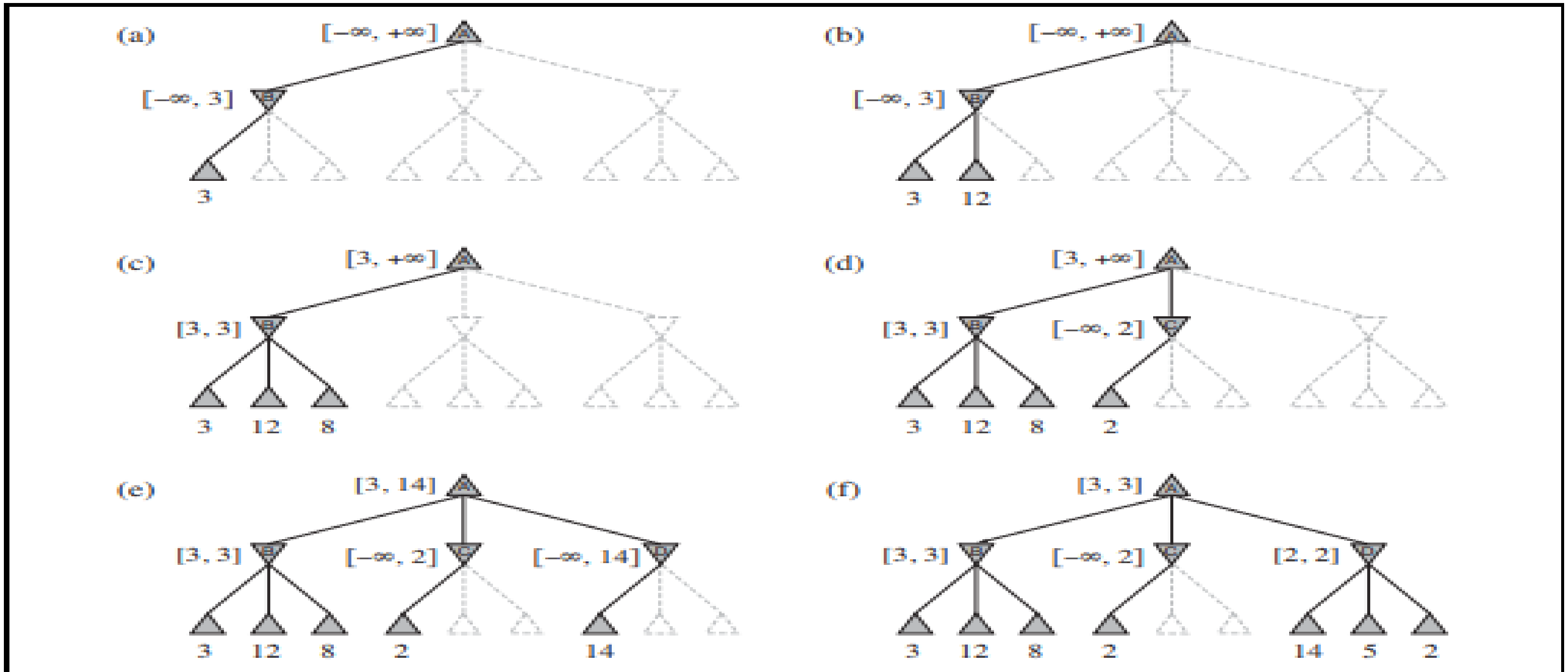

Figure Stages in the calculation of the optimal decision for the game tree in Figure 5.2. At each point, we show the range of possible values for each node. (a) The first leaf below *B* has the value 3. Hence, *B*, which is a MIN node, has a value of *at most* 3. (b) The second leaf below *B* has a value of 12; MIN would avoid this move, so the value of *B* is still at most 3. (c) The third leaf below *B* has a value of 8; we have seen all *B*'s successor states, so the value of *B* is exactly 3. Now, we can infer that the value of the root is *at least* 3, because MAX has a choice worth 3 at the root. (d) The first leaf below *C* has the value 2. Hence, *C*, which is a MIN node, has a value of *at most* 2. But we know that *B* is worth 3, so MAX would never choose *C*. Therefore, there is no point in looking at the other successor states of *C*. This is an example of alpha–beta pruning. (e) The first leaf below *D* has the value 14, so *D* is worth *at most* 14. This is still higher than MAX's best alternative (i.e., 3), so we need to keep exploring *D*'s successor states. Notice also that we now have bounds on all of the successors of the root, so the root's value is also at most 14. (f) The second successor of *D* is worth 5, so again we need to keep exploring. The third successor is worth 2, so now *D* is worth exactly 2. MAX's decision at the root is to move to *B*, giving a value of 3.

Alpha–beta Pruning

- Another way to look at this is as a simplification of the formula for MINIMAX. Let the two unevaluated successors of node C in Figure (Refer last figure) have values x and y.
- Then the value of the root node is given by

$$\begin{aligned}\textsc{Minimax}(root) &= \max(\min(3,12,8), \min(2,x,y), \min(14,5,2)) \\ &= \max(3, \min(2,x,y), 2) \\ &= \max(3, z, 2) \qquad \text{where } z = \min(2,x,y) \le 2 \\ &= 3.\end{aligned}$$

Alpha–beta Pruning

- In other words, the value of the root and hence the minimax decision are independent of the values of the pruned leaves x and y.
- Alpha–beta pruning can be applied to trees of any depth, and it is often possible to prune entire subtrees rather than just leaves.
- The general principle is this: consider a node n somewhere in the tree (Refer Figure P.T.O.), such that Player has a choice of moving to that node.

Alpha–beta Pruning

- If Player has a better choice m either at the parent node of n or at any choice point further up, then n will never be reached in actual play. So once we have found out enough about n (by examining some of its descendants) to reach this conclusion, we can prune it.
- Alpha–beta pruning gets its name from the following two parameters(Refer Fig) that describe bounds on the backed-up values that appear anywhere along the path.

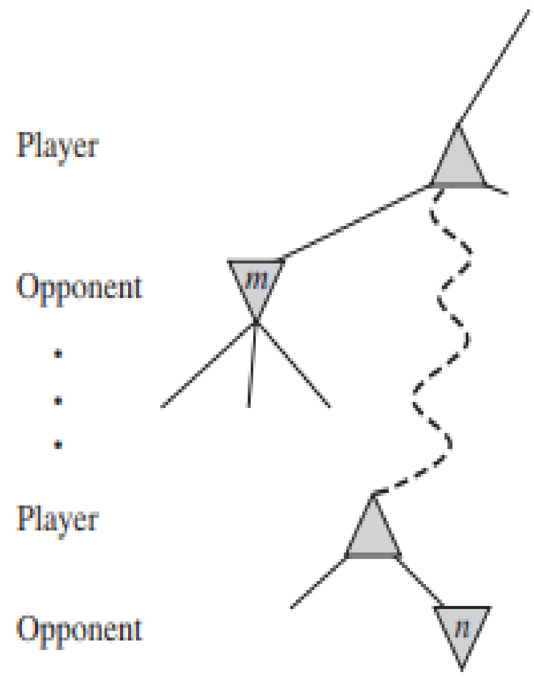

Figure The general case for alpha–beta pruning. If m is better than n for Player, we will never get to n in play.

$\alpha =$ the value of the best (i.e., highest-value) choice we have found so far at any choice point along the path for MAX.

$\beta =$ the value of the best (i.e., lowest-value) choice we have found so far at any choice point along the path for MIN.

Alpha–beta Pruning

- Alpha–beta search updates the values of α and β as it goes along and prunes the remaining branches at a node (i.e., terminates the recursive call) as soon as the value of the current node is known to be worse than the current α or β value for MAX or MIN, respectively.