

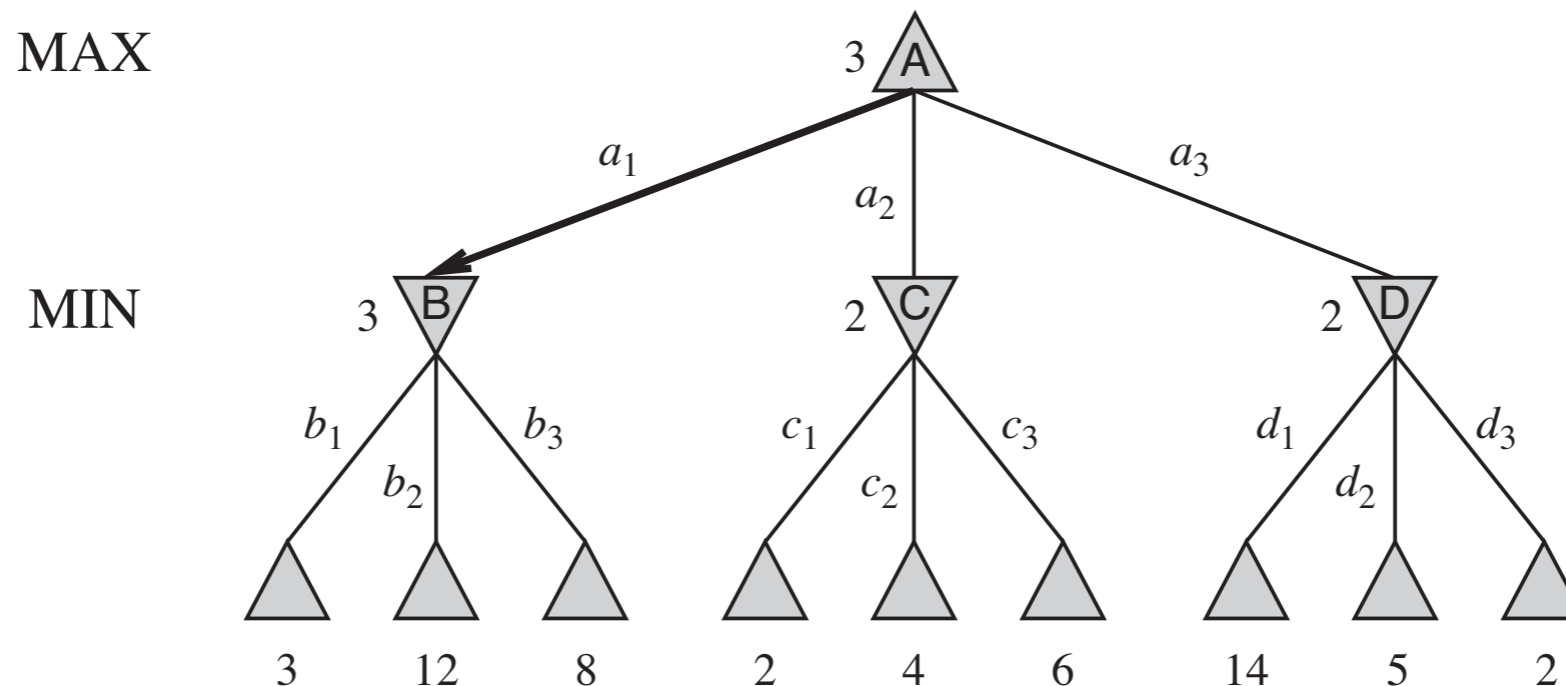
# Adversarial search

complementary slides

T. Svoboda, BE5B33KUI

2017-03-27

materials from AIMA 3rd edition



**Figure 5.2** A two-ply game tree. The  $\triangle$  nodes are “MAX nodes,” in which it is MAX’s turn to move, and the  $\nabla$  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.

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if } \text{TERMINAL-TEST}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if } \text{PLAYER}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if } \text{PLAYER}(s) = \text{MIN} \end{cases}$$

**function** MINIMAX-DECISION(*state*) **returns** *an action*  
**return**  $\arg \max_{a \in \text{ACTIONS}(s)} \text{MIN-VALUE}(\text{RESULT}(state, a))$

---

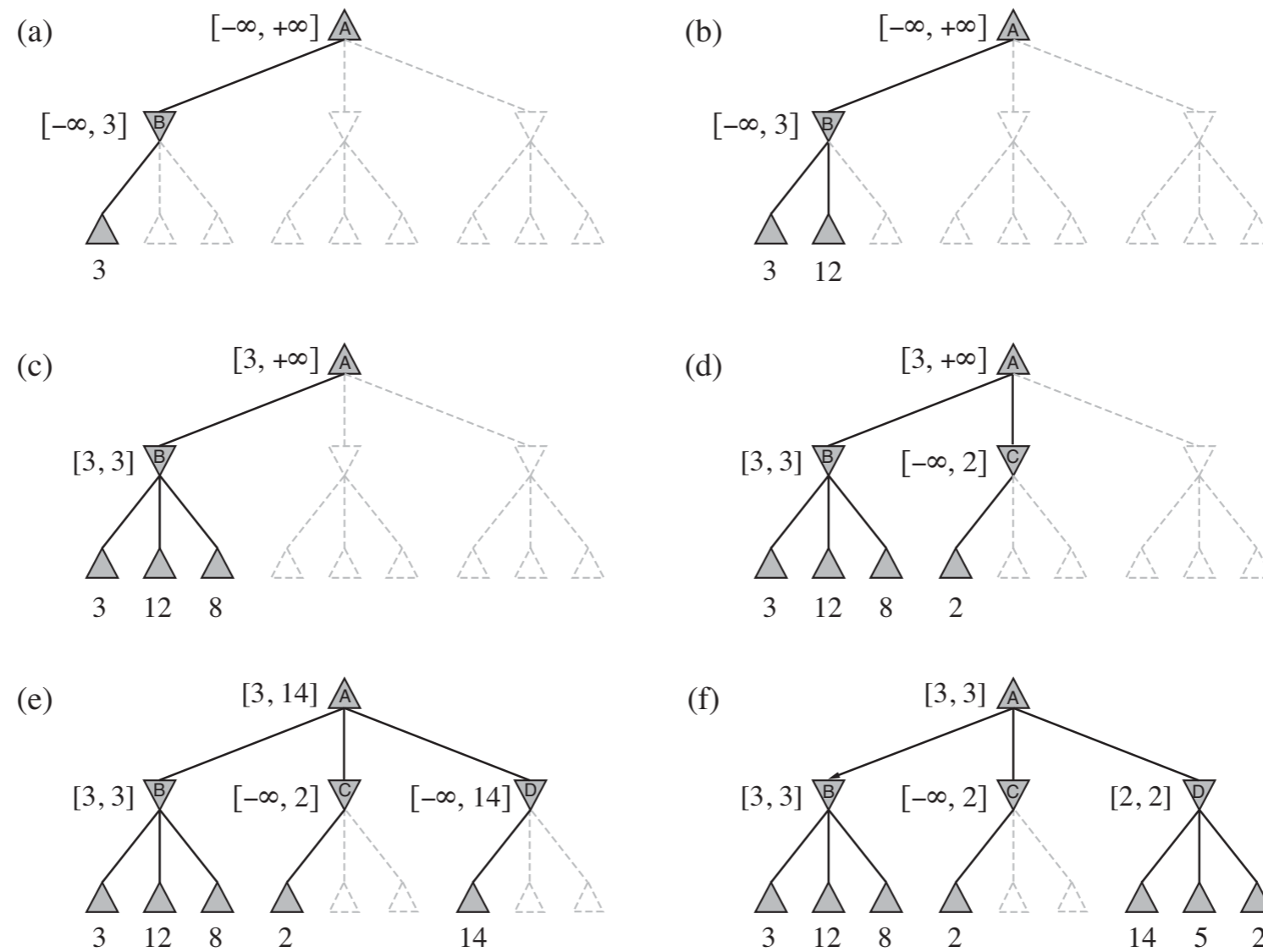
**function** MAX-VALUE(*state*) **returns** *a utility value*  
**if** TERMINAL-TEST(*state*) **then return** UTILITY(*state*)  
 $v \leftarrow -\infty$   
**for each** *a* **in** ACTIONS(*state*) **do**  
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$   
**return** *v*

---

**function** MIN-VALUE(*state*) **returns** *a utility value*  
**if** TERMINAL-TEST(*state*) **then return** UTILITY(*state*)  
 $v \leftarrow \infty$   
**for each** *a* **in** ACTIONS(*state*) **do**  
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$   
**return** *v*

---

**Figure 5.3** An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimize utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state. The notation  $\arg \max_{a \in S} f(a)$  computes the element *a* of set *S* that has the maximum value of *f(a)*.



**Figure 5.5** 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.

**function** ALPHA-BETA-SEARCH(*state*) **returns** an action  
 $v \leftarrow \text{MAX-VALUE}(\textit{state}, -\infty, +\infty)$   
**return** the *action* in ACTIONS(*state*) with value *v*

---

**function** MAX-VALUE(*state*,  $\alpha$ ,  $\beta$ ) **returns** a utility value  
**if** TERMINAL-TEST(*state*) **then return** UTILITY(*state*)  
 $v \leftarrow -\infty$   
**for each** *a* **in** ACTIONS(*state*) **do**  
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$   
    **if**  $v \geq \beta$  **then return** *v*  
     $\alpha \leftarrow \text{MAX}(\alpha, v)$   
**return** *v*

---

**function** MIN-VALUE(*state*,  $\alpha$ ,  $\beta$ ) **returns** a utility value  
**if** TERMINAL-TEST(*state*) **then return** UTILITY(*state*)  
 $v \leftarrow +\infty$   
**for each** *a* **in** ACTIONS(*state*) **do**  
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$   
    **if**  $v \leq \alpha$  **then return** *v*  
     $\beta \leftarrow \text{MIN}(\beta, v)$   
**return** *v*

---

**Figure 5.7** The alpha–beta search algorithm. Notice that these routines are the same as the MINIMAX functions in Figure 5.3, except for the two lines in each of MIN-VALUE and MAX-VALUE that maintain  $\alpha$  and  $\beta$  (and the bookkeeping to pass these parameters along).