A Formal Basis for the Heuristic Determination of Minimum Cost Paths

PETER E. HART, MEMBER, IEEE, NILS J. NILSSON, MEMBER, IEEE, AND BERTRAM RAPHAEL

Search Algorithm $A^*$:

1) Mark $s$ "open" and calculate $f(s)$.
2) Select the open node $n$ whose value of $f$ is smallest. Resolve ties arbitrarily, but always in favor of any node $n \in T$.
3) If $n \in T$, mark $n$ "closed" and terminate the algorithm.
4) Otherwise, mark $n$ closed and apply the successor operator $\Gamma$ to $n$. Calculate $\hat{f}$ for each successor of $n$ and mark as open each successor not already marked closed. Remark as open any closed node $n_i$ which is a successor of $n$ and for which $f(n_i)$ is smaller now than it was when $n_i$ was marked closed. Go to Step 2.

C. Proof of the Optimality of $A^*$

The next lemma makes the important observation about the operation of $A^*$ that, under the consistency assumption, if node $n$ is closed, then $\hat{g}(n) = g(n)$. This fact is important for two reasons. First, it is used in the proof of the theorem about the optimality of $A^*$ to follow, and second, it states that $A^*$ need never reopen a closed node. That is, if $A^*$ expands a node, then the optimal path to that node has already been found. Thus, in Step 4 of the algorithm $A^*$, the provision for reopening a closed node is vacuous and may be eliminated.
Adversarial Search & Pruning
Plan

- Minimax
- Pruning minimax search
Game Representation

- **zero-sum** games of **perfect information**
- Two players: MAX vs MIN
- Players alternate actions: state space transitions
Representation Elements

- **PLAYER(s):** which player chooses the action in state $s$
- **ACTIONS(s):** what actions are available from state $s$
- **RESULT(s,a):** the state that results from action $a$ in state $s$
- **TERMINAL-TEST(s):** whether state $s$ is a terminal state
- **UTILITY(s):** the value of state $s$, usually only if terminal
Minimax Strategy

• Choose best move assuming opponent plays *optimally*
  
• i.e., opponent also uses minimax

• MINIMAX(s) =
  
  if TERMINAL-TEST(s) then UTILITY(s)
  if PLAYER(s) = MAX then
    max of MINIMAX(RESULT(s,a)) for a in ACTIONS(s)
  if PLAYER(s) = MIN then
    min of MINIMAX(RESULT(s,a)) for a in ACTIONS(s)
\[ \text{MINIMAX}(s) = \]
\[
\text{if TERMINAL-TEST}(s) \text{ then UTILITY}(s) \\
\text{if PLAYER}(s) = \text{MAX} \text{ then} \\
\quad \text{max of } \text{MINIMAX}(\text{RESULT}(s,a)) \text{ for } a \text{ in ACTIONS}(s) \\
\text{if PLAYER}(s) = \text{MIN} \text{ then} \\
\quad \text{min of } \text{MINIMAX}(\text{RESULT}(s,a)) \text{ for } a \text{ in ACTIONS}(s) \]
**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 \( \text{MAX-VALUE} \) and \( \text{MIN-VALUE} \) go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state.

The notation \( \operatorname{argmax}_{a \in S} f(a) \) computes the element \( a \) of set \( S \) that has the maximum value of \( f(a) \).
What about the MINIMAX algorithm does not work if the game is *not* zero-sum?

How can we adjust the algorithm to address this problem?
For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will never be lower than the utility obtained against an optimal MIN.
Pruning

MAX

MIN
Alpha-Beta Pruning

- $[\alpha, \beta]$
  - $\alpha$ = upper-bound on minimax value
  - $\beta$ = lower-bound on minimax value
$$3 = 12$$

(d) 

$$[3, +\infty]$$

$$[3, 3]$$

$$[-\infty, 2]$$

3 12 8 2

(f) 

$$[3, 3]$$
(e)
function ALPHA-BETA-SEARCH(state) returns an action
  $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$
  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 $v$
Notes

- Transposition table: cache previously-seen states
- Maximum-depth heuristics
Reading

- Chapter 5 up to 5.3