Expectiminimax tree

From Wikipedia, the free encyclopedia

An **expectiminimax tree** is a specialized variation of a minimax game tree for use in artificial intelligence systems that play two-player zero-sum games such as backgammon, in which the outcome depends on a combination of the player's skill and chance elements such as dice rolls. In addition to "min" and "max" nodes of the traditional minimax tree, this variant has "chance" ("move by nature") nodes, which take the expected value of a random event occurring.[1] In game theory terms, an expectiminimax tree is the game tree of an extensive-form game of perfect, but incomplete information.

In the traditional minimax method, the levels of the tree alternate from max to min until the depth limit of the tree has been reached. In an expectiminimax tree, the "chance" nodes are interleaved with the max and min nodes. Instead of taking the max or min of the utility values of their children, chance nodes take a weighted average, with the weight being the probability that that child is reached.[1]

The interleaving depends on the game. Each "turn" of the game is evaluated as a "max" node (representing the AI player's turn), a "min" node (representing a potentially-optimal opponent's turn), or a "chance" node (representing a random effect or player).[1]

For example, consider a game in which each round consists of a single dice throw, and then decisions made by first the AI player, and then another intelligent opponent. The order of nodes in this game would alternate between "chance", "max" and then "min".[1]

Pseudocode [edit]

The expectiminimax algorithm is a variant of the minimax algorithm and was first proposed by Donald Michie.[2] Its pseudocode is given below.

```
function expectiminimax(node, depth)
    if node is a terminal node or depth = 0
        return the heuristic value of node
    if the adversary is to play at node
        // Return value of minimum-valued child node
        let α := +∞
        foreach child of node
            α := min(α, expectiminimax(child, depth-1))
    else if we are to play at node
        // Return value of maximum-valued child node
        let α := -∞
        foreach child of node
            α := max(α, expectiminimax(child, depth-1))
    else if random event at node
        // Return weighted average of all child nodes' values
        let α := 0
        foreach child of node
            α := α + (Probability[child] * expectiminimax(child, depth-1))
    return α
```

Note that for random nodes, there must be a known probability of reaching each child. (For most games of chance, child nodes will be equally-weighted, which means the return value can simply be the average of all child values.)

See also [edit]

- Minimax
- Expected value

References [edit]

1. ^ [a] [b] [c] [d] Stuart J. Russell; Peter Norvig (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall. pp. 177–178. ISBN 978-0-13-604259-4.

Wikipedia The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
Edit links

2. ^ D. Michie (1966). Game-playing and game-learning automata. In L. Fox (ed.), Advances in Programming and Non-Numerical Computation, pp. 183-200.

Categories: Game theory | Search algorithms | Game artificial intelligence | Trees (data structures)