Board Game Details

Project Type

Implementation of the Expectiminimax algorithm for a board game using two dice played by two players (human and computer).

Description

The board game consists of an array of N cells as shown below. The players take equal number of turns throwing two dice to reach the goal node or beyond, starting from 1. The board has a set of constraints, for example, if the person reaches 11, he leaps to position 15. Likewise, if he reaches 18, he is taken backward to 14. The board can have any number of constraints like this [1]. The player who first reaches the goal node or beyond wins. If both the players reach the same tile at or beyond the goal node, then the game is a draw. For each turn, the players have six possible actions:

- Move Forward: Dice 1 rolled value / Dice 2 rolled value / Dice 1 + Dice 2 rolled value
- Move Backward: Dice 1 rolled value / Dice 2 rolled value / Dice 1 + Dice 2 rolled value

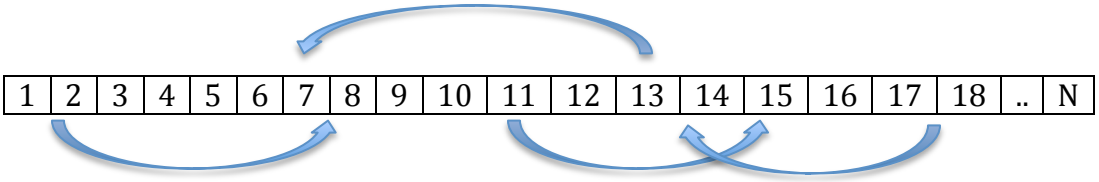


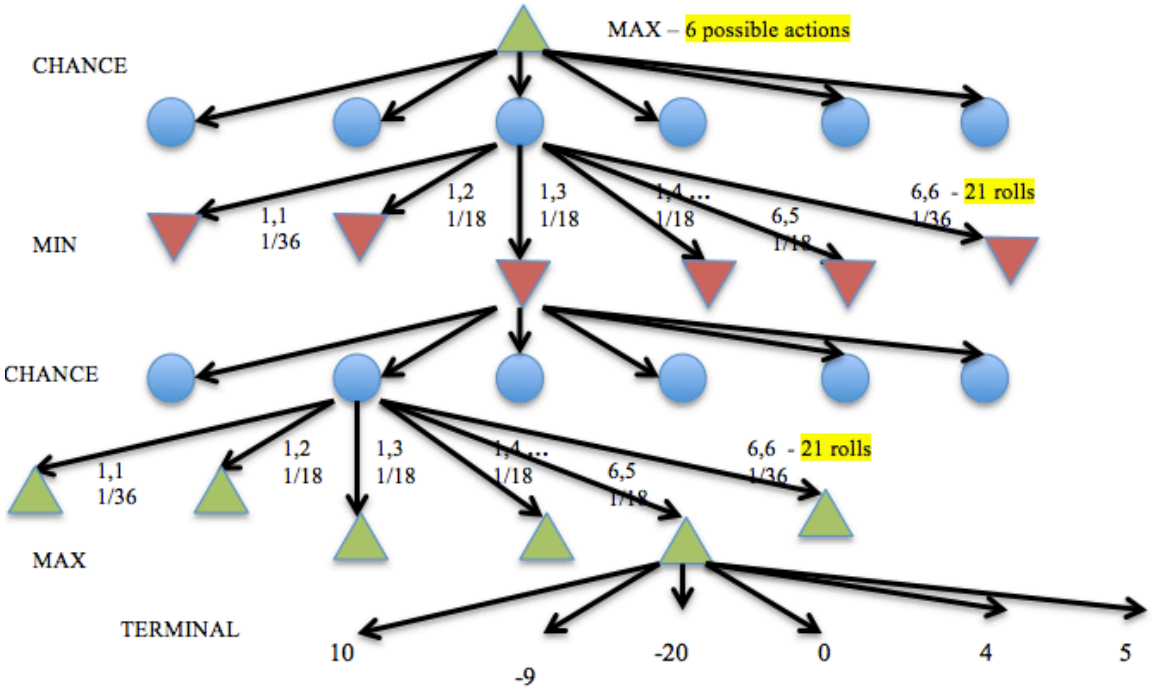
Figure 1: Board Game of N cells with constraints

Since this type of a game involves unpredicatability because it depends on the dice rolls, it includes chance nodes in addition to Min and Max nodes. Here, we calculate the weighted average where the weight is the probability of reaching the child. [2, 3]

```
// Return weighted average of all child nodes' values
let α := 0
foreach child of node
 α := α + (Probability[child] * expectiminimax(child, depth-1))
```

Figure 2: Chance Node Calculation [2]

Game Tree for the Board Game with a max depth of 5



- Max has 6 possible actions for a dice roll – Move forward/backward for either die roll 1 or die roll 2 or die roll 1+2 values. It will pick the maximizing action.
- It calls chance node, which computes the weighted average of 21 rolls by multiplying each rolls probability (1/18 or 1/36) with value of successive min node.
- Each Min player will in-turn have 6 possible actions and it will pick the minimizing action for the Max player.
- The game tree follows the same pattern until it sees a terminal node or if max depth is reached: Max -> Chance -> Min -> Chance -> … -> Terminal.

References:

[1] CS3600 – Introduction to Artificial Intelligence, Advesarial Games with Dice, from: http://www.cc.gatech.edu/~riedl/classes/2013/cs3600/homeworks/chutes-soln.pdf [Accessed: 06-Apr- 2016]

[2] Expectiminimax Tree, from: https://en.wikipedia.org/wiki/Expectiminimax_tree [Accessed: 06-Apr- 2016]

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