

Parallel Functional Programming Project Proposal

Using Parallel Alpha-Beta Pruning to Solve 2048

Unal Yigit Ozulku (UNI: uyo2000)

2048:

“2048 is played on a plain 4×4 grid, with numbered tiles that slide when a player moves them using the four arrow keys. Every turn, a new tile randomly appears in an empty spot on the board with a value of either 2 or 4. Tiles slide as far as possible in the chosen direction until they are stopped by either another tile or the edge of the grid. If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided. The resulting tile cannot merge with another tile again in the same move.” (Wikipedia)

Minimax:

“Minimax is a recursive algorithm which is used to choose an optimal move for a player assuming that the adversary is also playing optimally” (Medium). This project will focus on parallelizing the minimax algorithm in Haskell and analyzing metrics to understand the difference in performance from a sequential implementation to a parallel one.

Minimax Search Algorithm

```
function MINIMAX-SEARCH(game, state) returns an action
  player ← game.TO-MOVE(state)
  value, move ← MAX-VALUE(game, state)    Assuming root is MAX
  return move

function MAX-VALUE(game, state) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
  v ← -∞
  for each a in game.ACTIONS(state) do
    v2, a2 ← MIN-VALUE(game, game.RESULT(state, a))    MAX calls MIN
    if v2 > v then
      v, move ← v2, a
  return v, move

function MIN-VALUE(game, state) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
  v ← +∞
  for each a in game.ACTIONS(state) do
    v2, a2 ← MAX-VALUE(game, game.RESULT(state, a))    MIN calls MAX
    if v2 < v then
      v, move ← v2, a
  return v, move
```

Picture from COMS 4701 lecture slides

Alpha-beta pruning:

“Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It stops evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.” (Wikipedia)

General idea: Keep track of highest (α) and lowest (β) values seen so far by MAX and MIN nodes, respectively

- Skip remaining children (prune) if:
- MAX sees value higher than β
- MIN sees value lower than α

```


### Alpha-Beta Search



---



```

function ALPHA-BETA-SEARCH(game, state) returns an action
 player ← game.TO-MOVE(state)
 value, move ← MAX-VALUE(game, state, -∞, +∞) Assuming root is MAX
 return move

function MAX-VALUE(game, state, α, β) returns a (utility, move) pair
 if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
 v ← -∞
 for each a in game.ACTIONS(state) do
 v2, a2 ← MIN-VALUE(game, game.RESULT(state, a), α, β)
 if v2 > v then
 v, move ← v2, a
 α ← MAX(α, v)
 if v ≥ β then return v, move MAX updates α, compares against β
 return v, move

function MIN-VALUE(game, state, α, β) returns a (utility, move) pair
 if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
 v ← +∞
 for each a in game.ACTIONS(state) do
 v2, a2 ← MAX-VALUE(game, game.RESULT(state, a), α, β)
 if v2 < v then
 v, move ← v2, a
 β ← MIN(β, v)
 if v ≤ α then return v, move MIN updates β, compares against α
 return v, move

```


```

Picture from COMS 4701 lecture slides

Sources:

- [https://en.wikipedia.org/wiki/2048_\(video_game\)](https://en.wikipedia.org/wiki/2048_(video_game))
- COMS 4701 – Artificial Intelligence (Columbia University, Prof. Tony Dear, Summer 2021)

- <https://medium.com/@bartoszzadrony/beginners-guide-to-ai-and-writing-your-own-bot-for-the-2048-game-4b8083faaf53>
- https://en.wikipedia.org/wiki/Alpha-beta_pruning