Universidade Federal de Uberlândia

Algoritmo MiniMax e Poda Alfa-Beta

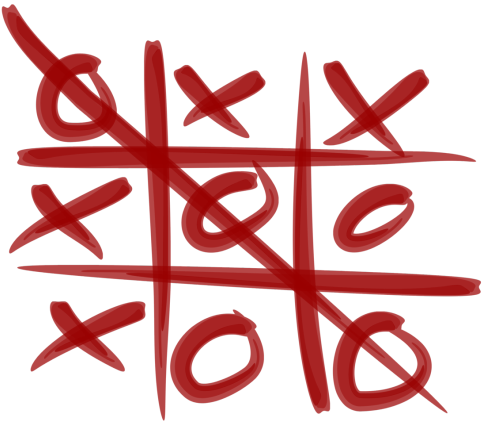
Higor Lucas De Araújo Freitas

Professor: Márcia Aparecida Fernandes

Outubro de 2023

Introdução

- Jogo da Velha
  - Introdução
  - Algoritmo miniMax
  - Algoritmo Alfa-Beta
  - Implementação Codigo



Introdução

- O jogo da velha é um jogo de matriz 3 por 3 onde se joga com dois jogadores um será o X o outro será o O que a gente chama de círculo ou bolinha, objetivo dos jogadores é fazer um sequência de 3 símbolos iguais, onde pode ser feito em linhas ou em diagonais.

Introdução: Objetivo

- A proposta deste estudo é utilizar o Algoritmo MiniMax e o Algoritmo de Poda Alfa-Beta para comparar quais são as melhores abordagens para utilizar uma o algoritmo de busca, onde teremos dois jogadores, escolhemos utilizar o jogo da Velha por ser autodidata.

Algoritmo MiniMax

- O algoritmo MiniMax é um método para minimizar a possível perda máxima. Pode ser considerado como a maximização do ganho mínimo (maximin). Com tudo irá abrir uma árvore de possibilidades onde ele entrará nos ramos, buscando valores, esses valores será denominado a cada recursividade entrada no algoritmo, a cada fase ele irá analisar se é o mínimo que ele precisa para maximizar o máximo que irá precisar. E assim por diante até atender a condição de parada da chamada recursiva.

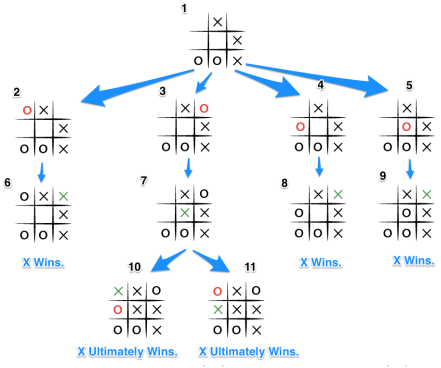


Figura: Sequência das jogadas

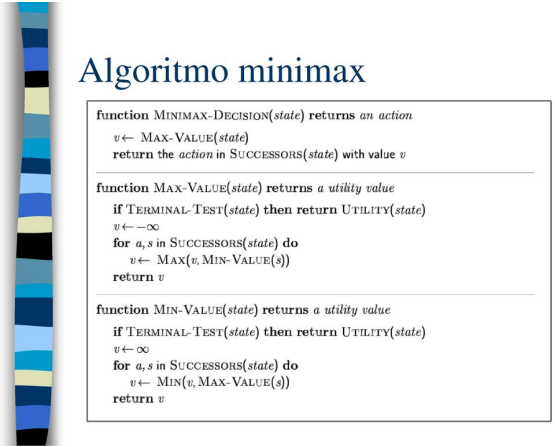


Figura: Algoritmo MiniMax

Algoritmo Poda Alfa-Beta

- O algoritmo Poda Alfa-Beta utiliza as mesmas análise do MiniMax porém ele vai remover os ramos que não vão ser necessários, analisando em vez de entrar em todas as possibilidades ele vai verificar se o alfa é maior ou igual o beta no min e no max, assim matando o ramo que não precisa ser aberto, maximizando os nó da árvore.

Algoritmo Poda Alfa-Beta

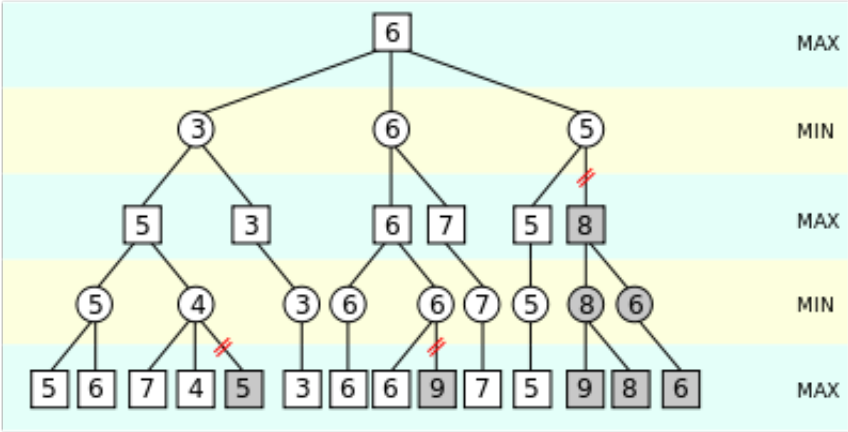


Figura: Sequência das jogadas

Algoritmo Poda Alfa-Beta

```
export const minimax_alfa_beta = (
  board,
  jogadorNovo,
  jogadorAntigo,
  alpha,
  beta
) => {
  let parada = potoDeParada(board,jogadorAntigo);
  if(parada != null){
    return parada;
  }

  let jogadas = jogadasPossiveis(board);

  resultComparacao.qtd += 1;
  resultComparacao.player = jogadorNovo;

  if (jogadorNovo == jogadorAntigo) {
    let bestScore = -10;
    for (let i in jogadas) {
      let resultado = jogada(board, jogadas[i], jogadorNovo);
      let valor = minimax_alfa_beta(
        resultado,
        jogadorNovo == "X" ? "O" : "X",
        jogadorAntigo,
        alpha,
        beta
      );
      bestScore = Math.max(bestScore, valor);
      alpha = Math.max(alpha, bestScore);
      if (beta <= alpha) {
        break;
      }
    }
    return bestScore;
  } else {
    let bestScore = 10;
    for (let i in jogadas) {
      let resultado = jogada(board, jogadas[i], jogadorNovo);
      let valor = minimax_alfa_beta(
        resultado,
        jogadorNovo == "X" ? "O" : "X",
        jogadorAntigo,
        alpha,
        beta
      );
      bestScore = Math.min(bestScore, valor);
      beta = Math.min(beta, bestScore);
      if (beta <= alpha) {
        break;
      }
```

Figura: Algoritmo Poda Alfa-Beta

Experimentos: Características do modelo

- Algoritmo MiniMax vs Poda Alfa-Beta
  - Iniciando as Etapas(8) o Algoritmo MiniMax abre 295137 nós enquanto o Alfa-beta abre 18168 diferença de 16x entre os dois
  - Velocidade de execução de velocidade entre eles é de 909 para o Minimax contra 82 milesegundos do Alfa-Beta
  - Todos Obtiveram os mesmo resultado, Empate idependente da onde foi iniciado

Resultados

MiniMax				Poda Alfa-Beta			
Ramos	Nós	Porcentagem 9! Possibilidades	Tempo	Ramos	Nós	Porcentagem 9! Possibilidades	Tempo
8	295137	81,33%	729	8	18168	5,01%	68
7	31996	8,82%	142	7	2525	0,70%	8
6	3871	1,07%	31	6	904	0,25%	4
5	477	0,13%	5	5	137	0,04%	1
4	103	0,03%	1	4	55	0,02%	1
3	25	0,01%	1	3	19	0,01%	0
2	7	0,00%	0	2	7	0,00%	0
1	2	0,00%	0	1	2	0,00%	0
SOMA	331618		909		21817		82

Figura: Tabela de resultados

Resultados

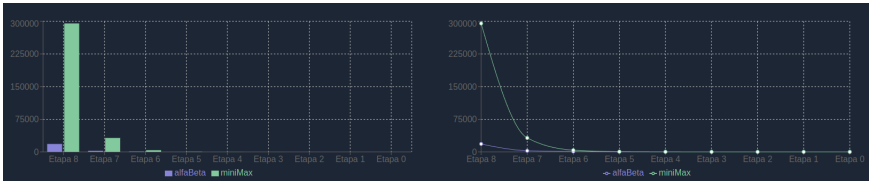


Figura: Analise do Grafico comprando etapas

Algoritmo - Explicação

- O algoritmo começa com o Best, onde será selecionado o melhor nó, isso vai nos retornar as posições que foi a melhor do nosso algoritmo minimax.

```
export const best = (board, jogadorAntigo, algoritmo) => {
  let jogadas = jogadasPossiveis(board);
  let best = -10;
  let posicaoJogada = null;
  resultComparacao.qtd = 0;
  let execution = new Date();

  resultComparacao.algoritmo = algoritmo.name;
  for (let i in jogadas) {
    let resultado = jogada(board, jogadas[i], jogadorAntigo);
    let valor = algoritmo(
      resultado,
      jogadorAntigo == "X" ? "O" : "X",
      jogadorAntigo,
      -10,
      10
    );
    //console.log(jogadas[i], i);
    //console.log(valor);
    resultComparacao.node = i;
    if (valor > best) {
      best = valor;
      posicaoJogada = jogadas[i];
      resultComparacao.row = posicaoJogada[0];
      resultComparacao.columns = posicaoJogada[1];
    }
    //console.log(resultado);
  }

  resultComparacao.time =  new Date() - execution + " ms";

  //console.log(resultComparacao)

  return [posicaoJogada,resultComparacao];
  // return posicaoJogada;
};
```

Figura: Best busca posição para jogar

Algoritmo - Explicação

- No entorno do best temos as jogadas possiveis que vai retornar as posições onde podemos colocar nosso mini ou max.

```
const jogadasPossiveis = (board) ⇒ {
  var jogadas = [];
  for (var i = 0; i < 3; i++) {
    for (var j = 0; j < 3; j++) {
      if (board[i][j] == "") jogadas.push([i, j]);
    }
  }
  return jogadas;
};
```

- Logo a baixo temos o best que será inicializado com um valor negativo como -10 depois a posição para jogar em seguida vamos interar o jogadas para poder assim jogar, entrando nele tempos o jogadar

```
const jogada = (board, jogadorNova, jogadorAntigo) ⇒ {
  let copyBoard = deepCopy(board);
  copyBoard[jogadorNova[0]][jogadorNova[1]] = jogadorAntigo;
  return copyBoard;
};
```

Algoritmo - Explicação

- Logo apos fazer a jogada ele entra dentro do algoritmo que pode ser tanto o minimax quanto o poda alfa-beta

```
export const minimax = (board, jogadorNova, jogadorAntigo) => {

  let parada = potoDeParada(board,jogadorAntigo);
  if(parada != null){
    return parada;
  }

  // Codigo de recusividade aqui
  let jogadas = jogadasPossiveis(board);

  resultComparacao.qtd += 1;
  resultComparacao.player = jogadorNova;
  // Significa que ele é o X
  if (jogadorNova == jogadorAntigo) {
    //MAX
    let best = -10;
    for (let i in jogadas) {
      let resultado = jogada(board, jogadas[i], jogadorNova);
      let valor = minimax(
        resultado,
        jogadorNova == "X" ? "O" : "X",
        jogadorAntigo
      );
      if (valor > best) {
        best = valor;
      }
    }

    return best;
  } else {
```

Algoritmo - Explicação

- Como nossa interação é sempre auto incremental ela precisa de um ponto de parada e por isso temos o potoDeParada

```
const potoDeParada = (board,jogadorAntigo) ⇒{
  let w = verificarQuemGanhou(board);
  if (w == jogadorAntigo) return 1;
  if (w && w ≠ jogadorAntigo) return -1;
  if (!w && verificarSeEmpatou(board)) return 0;
}
```

- Assim que ele inicializa nosso estado ele vai analisando se é o mini ou max entrando recursivamente até finalizar.

Algoritmo Poda Alfa-Beta

```
export const minimax_alfa_beta = (
  board,
  jogadorNovo,
  jogadorAntigo,
  alpha,
  beta
) => {
  let parada = potoDeParada(board,jogadorAntigo);
  if(parada != null){
    return parada;
  }

  let jogadas = jogadasPossiveis(board);

  resultComparacao.qtd += 1;
  resultComparacao.player = jogadorNovo;

  if (jogadorNovo == jogadorAntigo) {
    let bestScore = -10;
    for (let i in jogadas) {
      let resultado = jogada(board, jogadas[i], jogadorNovo);
      let valor = minimax_alfa_beta(
        resultado,
        jogadorNovo == "X" ? "O" : "X",
        jogadorAntigo,
        alpha,
        beta
      );
      bestScore = Math.max(bestScore, valor);
      alpha = Math.max(alpha, bestScore);
      if (beta <= alpha) {
        break;
      }
    }
    return bestScore;
  } else {
    let bestScore = 10;
    for (let i in jogadas) {
      let resultado = jogada(board, jogadas[i], jogadorNovo);
      let valor = minimax_alfa_beta(
        resultado,
        jogadorNovo == "X" ? "O" : "X",
        jogadorAntigo,
        alpha,
        beta
      );
      bestScore = Math.min(bestScore, valor);
      beta = Math.min(beta, bestScore);
      if (beta <= alpha) {
        break;
      }
```

Figura: Algoritmo Poda Alfa-Beta

Conclusões

- O objetivo de aplicação do Algoritmo MiniMax e Poda Alfa Beta no tema deste trabalho foi alcançado visto que o modelo aplicado atingiu uma porcentagem com comparação a quantidade de nós gerados, onde o MiniMax no primeiro ramo gera cerca de 81% do total que é 362880 e o Alfa Beta chega a regarar 5% isso significa que a poda é a melhor forma de fazer com que podemos agilizar os processos.