# Jogo

# ­­­­­­­­­­­­­­­­­­­­de

# Dominó

# Integrantes

|  |  |
| --- | --- |
| Matrícula | Nome |
| 112031005 | André Alvarado |
| 212031119 | Erick Valadares |
| 212031132 | Daniel Derlando |
| 212031098 | Lucas Tito |
| 113031049 | Matheus Filipe |

Sumário

1 Introdução

2 API e tecnologias

2.1 Instruções

3 Ambiente

4 diagrama de classes

5 Métodos de Inteligência Artificial

5.1 MinMax com representação de conhecimento

5.2 minmax alfa-beta

6 Resultados

7 Conclusão e observações

# 1 Introdução

O jogo escolhido para este trabalho é o clássico Dominó, em uma versão onde cada jogador recebe 14 peças, ou seja, não existe mercado para compra.

Os algoritmos de inteligência artificial usados neste trabalho foram minmax poda alfa-beta e minmax poda alfa-beta cut-off, que serão melhor descritos neste documento, na seção 4.

**Regras do jogo:**

1 - O jogo inicia com o jogador que possui o gabão de 6 (gabão é uma peça que possui os dois lados iguais)

2 - Em um turno, somente um jogador pode jogar

3 - Ao final de um turno, inicia-se outro turno, dando a vez de jogo para o próximo jogador.

4 - Uma peça é jogável, quando um de seus lados possui o mesmo valor do lado sem conexão, de uma das peças da extremidade.

5 - É dita uma conexão de peças, quando uma peça está encostada em outra peça e os lados das peças que se encostam possuem o mesmo valor. Ex: [6/6][6/4][4/2] possui 2 conexões.

6 - Um jogador pode não jogar, passar a vez, quando ele não tiver uma peça, que é jogável dado o estado da mesa.

7 - Cada jogador começa com 14 peças

8 - O jogador que acabar primeiro com todas as suas peças, ganha o jogo

9 - Se nenhum jogador tiver peças a jogar e os mesmos tiverem peça (mesa fechada), ganha o jogo quem tiver menos pontos.

10 - A contagem de pontos é feita da seguinte maneira:

Somatório de 0 a n (lado esquerdo + lado direito), onde n é a quantidade de peças de um jogador.

11 - Quando um jogador começa o jogo com mais de 4 gabões, a redistribuição de peças para ambos os jogadores deverá ser feita (Arria a mão).

# 2 API e tecnologias

Linguagem: Java (JDK 8 update 20)

IDE: Netbeans 8.0.1 Full

Componente gráfico: console

# 2.1 instruções

1 - Importar o projeto para o Netbeans

2 - Executar o projeto

3 - Jogar seguindo as regras do jogo descritas neste documento e as instruções do jogo.

# 3 Ambientes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Completamente observável | Multi agente | Estático | Sequencial | Determinístico |

O ambiente é completamente observável, visto que os jogadores podem saber as peças que o seus adversários possuem, com as informações de quais são as peças que estão na mesa e em sua mão de jogo.

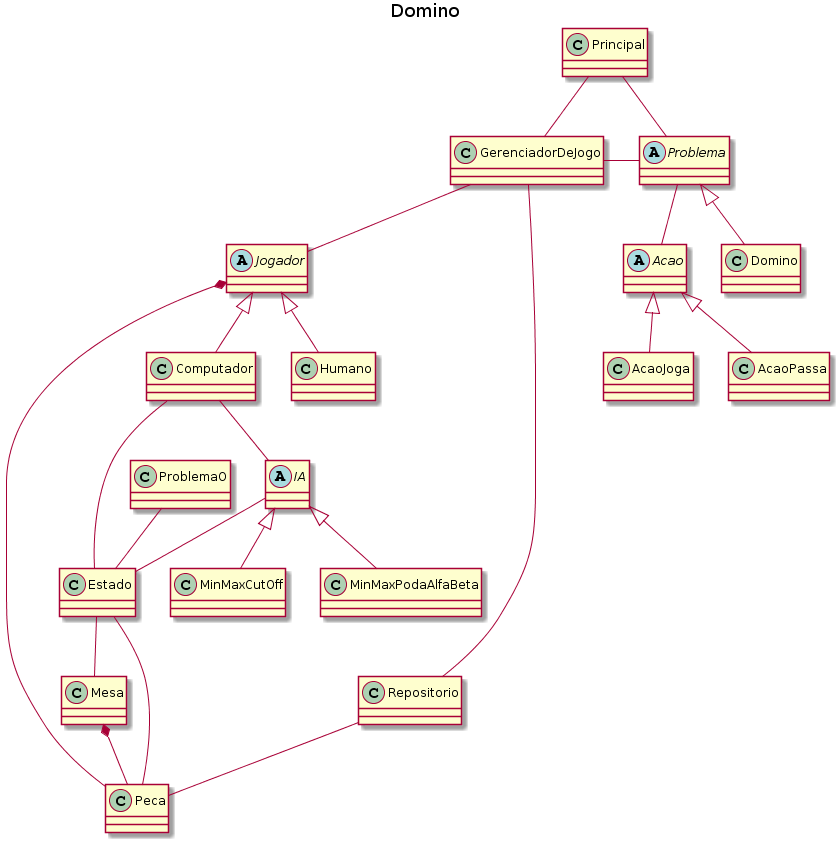
O dominó é multi agente, dado que existem dois jogadores (humano x computador) ou (computador x computador).

Estático, porque no turno de um jogador, só ele pode jogar e o estado de jogo será alterado depois de sua jogada.

Sequencial, porque uma jogada interfere nas escolhas futuras.

Entendemos que o jogo de dominó é determinístico, porque dado uma jogada, o estado é bem definido e independe de probabilidade, mesmo que possa ser interpretada a probabilidade de um jogador jogar uma determinada peça.

# 4 Diagrama de classes



# 5 Métodos de Inteligência Artificial

# 5.1 Minmax poda alfa-beta

O método de poda alfa-beta é como descrito na disciplina e sua função de utilidade é definida por:

1 Integer.maxvalue quando a IA ganha

2 Integer.minvalue quando a IA perde

3 f(inimigo, IA) = (mão(Inimigo) – mão(IA)) quando o jogo empata. Se a mão da IA possuir mais pontos, o valor da função será negativo, indicando que a IA perdeu, Se o valor da função for positivo, significa que a mão da IA possui menos pontos, levando a vitória.

# 5.2 Minmax poda alfa-beta cut-off

O método de poda alfa-beta com cut-off é como descrito na disciplina, tendo sua função de avaliação (eval) descrita abaixo.

Eval(estado) = f(estado) + g(estado) + h(estado) + i(estado) + j(estado) + k(estado)

Eval, então é uma função que basea-se na combinação linear das funções descritas abaixo.

|  |  |  |
| --- | --- | --- |
| Função | Peso | Descrição |
| F(estado) | 2 | Calcula quantas peças o inimigo poderá realizar dado um estado, quanto menor, melhor |
| g(estado) | 2 | Calcula qual quantidade de gabões podem ser descartados dado o estado (quanto maior, melhor) |
| h(estado) | 1 | Calcula quantos gabões a mão do jogador terá (quanto menor, melhor) |
| i(estado) | 1 | Calcula a diferença entre a quantidade das peças do inimigo e da IA (quanto maior o valor, melhor) |
| j(estado) | 2 | Calcula quantidade de pontas jogáveis (quanto mais melhor) |
| k(estado) | 2 | Calcula a quantidade de pontos do jogador (quanto mais, melhor) [inimigo - IA] |

# 6 Resultados

Os métodos de inteligência artificial são influenciados pela distribuição das peças que é feita de forma aleatória. Uma “mão” pode ajudar ou atrapalhar o raciocínio da IA, ou seja, dada uma organização de peças uma IA tem maior probabilidade de ganhar que a outra.

Em geral, o algoritmo de poda alfa-beta com cut-off é mais rápido que o algoritmo de poda alfa-beta clássico, porém o algoritmo de poda alfa-beta ganhou mais vezes.

Abaixo segue uma tabela esperimental (não satisfatória, devido a possuir só 2 amostras por método e profundidade), a linha representa o algoritmo e a coluna a profundidade, a célula representa o tempo que a IA demorou em média para jogar (segundos).

|  |  |  |  |
| --- | --- | --- | --- |
|  | 15 | 10 | 5 |
| alfa-beta | 5,674 | 1,948 | 8,481 |
| cut-off | 1,326 | 0,738 | 0,003 |

# 7 Conclusão e Observações

Observou-se que o algoritmo minmax poda alfa-beta, tem um tempo de execução significativamente menor do que o algoritmo minmax tradicional e que o poda alfa-beta com cut-off possui um tempo de jogada menor que o de poda alfa-beta.

O algoritmo de minmax poda alfa-beta cut-off é satisfatório para nosso jogo, sendo este mais rápido e mais parecido com a jogada de um humano, permitindo que um jogador humano também ganhe.