Procura Adversarial:

1. Classificação e Cenários de jogo:

- Determinístico → Resultados/Ações previsíveis (xadrez)
- Não-Determinístico → Resultados/Ações mais aleatórios (Poker, Monopólio)

(Xadrez, Monopólio) - ° Info. Perfeita → Os jogadores conhecem o estado total do jogo

(Poker) - ° Info. Imperfeita → Os jogadores não tem informação total do estado do jogo.

2. Estratégias em Jogos:

2.1. Estratégias dominantes:

Uma estratégia dominante é aquela que é sempre a melhor escolha para um jogador, independentemente das escolhas do adversário.

2.2. Minimax e Ponto de Equilíbrio:

- Minimax → Maximizar o ganho em função de escolhas racionais assumindo que o adversário fará a jogada que minimize o seu ganho.
- Ponto de Equilíbrio → Se um jogo de soma 0, envolvendo dois jogadores, tem pontos de equilíbrio, portanto, o melhor que um jogador pode fazer (assumindo que ambos são racionais, é escolher a estratégia que contém um ponto de equilíbrio.
- Teorema de Von Neumann → Existe uma estratégia ótima para jogos de soma zero finito.

3. Cooperação e Competição:

Nem todos os cenários são meramente competitivos, a cooperação também é um tema importante.

- O Dois irmãos e o bolo → Um corta o bolo aproximadamente ao meio e o outro escolhe a fatia maior, incentivando a maximização do ganho de ambos. (1 corta o bolo o mais ao meio possível pq vai ficar sempre c/ a pior fatia).

4. Modelagem e Algoritmos para Procura em jogos:

A procura adversarial é modelada como uma árvore de estados.

- Estados: iniciais, finais
- Operadores: Movimentos que levam de um estado ao outro.
- Árvore de Procura: Representação dos possíveis estados e movimentos
- Função Recompensa: associada aos nós finais, avalia a utilidade de um estado terminal.
- Jogadores: Dois jogadores Max e Min. Max joga primeiro. Max tenta maximizar a função de recompesa, enquanto Min tenta minimiza-la, usando a mesma estratégia de avaliação.
- Retropropagação: Os valores recompensa dos nós finais são propagados de volta pela árvore para determinar os valores dos nós não finais.

MiniMax

```
função MiniMax(estado): movimento
  1. v← val_MAX(estado)
  2. devolve movimento associado ao sucessor de estado de valor v
fim_de_função
```

```
função val_MAX(estado): recompensa
  1. se final(estado) então
       1.1. devolve recompensa(estado)
     fim_de_se
  2. v ← -∞
  3. para s ∈ sucessores(estado) faz
       3.1. V ← max(v,val_MIN(s))
     fim_de_para
  4. devolve v
fim_de_função
```

```
função val_MIN(estado): recompensa
  1. se final(estado) então
       1.1. devolve recompensa(estado)
     fim_de_se
  2. v ← ∞
  3. para s ∈ sucessores(estado) faz
       3.1. V ← min(v,val_MAX(s))
     fim_de_para
  4. devolve v
fim_de_função
```

4.1. Corte Alfa-Beta:

↳ Igual a MiniMax mas reduz o número de nós a serem explorados na árvore de procura, eliminando ramos da árvore que não podem influenciar a decisão final, uma vez que se saiba que o jogador não escolherá esse ramo.

```
função alfa_beta(estado): movimento
  1. v ← val_MAX(estado,-∞, +∞)
  2. devolve movimento associado ao sucessor de estado de valor v
fim_de_função
```

```
função val_MAX(estado,alfa,beta): recomp.
  1. se final(estado) então
       1.1. devolve recompensa(estado)
  2. v ← -∞
  3. para s ∈ sucessores(estado) faz
       3.1. v ← max(v,val_MIN(s,alfa,beta))
       3.2. se v ≥β então
              3.2.1. devolve v
            fim_de_se
       3.3. α ← max(α,v)
     fim_de_para
  4. devolve v
fim_de_função
```

```
função val_MIN(estado,alfa,beta): recomp.
  1. se final(estado) então
       1.1. devolve recompensa(estado)
  2. v ← +∞
  3. para s ∈ sucessores(estado) faz
       3.1. v ← min(v,val_MAX(s,alfa,beta))
       3.2. se v ≤α então
              3.2.1. devolve v
            fim_de_se
       3.3. β ← min(β,v)
     fim_de_para
  4. devolve v
fim_de_função
```

5. Limitações e Soluções

A procura adversarial exaustiva (Minimax) tem limitações especialmente em jogos complexos:

- Limitações → Não é possível expandir a árvore de procura até ao fim em jogos com muitos estados.
- Assume-se que o adversário faz sempre a escolha ótima. O que pode não ser verdade em todos os casos.
- Usar o Expectimax (para jogos não determinístico, onde se considera a probabilidade dos resultados).
- Soluções: usar uma função de avaliação dos estados (em vez de expandir a árvore até ao fim, avalia-se a utilidade de um estado intermédio.

6. Funções de Avaliação:

↳ Dá uma estimativa de utilidade de um estado.

- ser exata na avaliação dos nós finais
- ser rápida
- estimar com a maior exatidão possível a pontuação associada a um nó não final.