Análise de Estratégias de Busca em IA

Igor Pereira dos Santos

igor.santos@utp.br

1 Introdução

Essa pesquisa tem como objetivo analisar mais afundo as estrategias de buscas utilizadas no âmbito da Inteligência Artificial, nosso objetivo e destacar a analise de alguns problemas tais eles como Cubo de Rubik, Missionários e canibais, Problema das n rainhas e Sudoku, utilizando estratégias de buscas com informação, sem informação e busca local baseado em agente.

2 Fundamentação Teórica

No âmbito das analises de estratégia de busca, possuímos diversos algoritmos que podemos utilizar para resolvermos um problema, temos os algoritmos de busca sem informação que são o algoritmo de busca em largura, busca em profundidade, busca por custo uniforme, busca em profundidade limitada, esses algoritmos utilizam se um estado inicial x e a partir de suas ações geram sucessores até encontrar o estado final.

Temos também os algoritmos de busca com informação que são o algoritmo de busca gulosa e o algoritmo A* que utilizam de informações adicionais para auxiliar em um problema onde não se busca o melhor custo de caminho e sim solucionar o problema.

Possuímos também os algoritmos de buscas locais e busca competitiva, na busca local temos o hill climbing e o simulated annealing, na busca competitiva utilizamos os algoritmos de minimização e maximização.

Cada um desses algoritmos podem ser utilizados para resolver um problema de busca, na nossa analise de problemas em questão, utilizaremos o algoritmo de busca com informação A*, utilizaremos o algoritmo de busca sem informação em profundidade e o algoritmo de busca local Hill Climbing.

2.1 Componentes

Basicamente partiremos do principio de uma resolução de problema por meio de busca, onde um problema pode ser definido por 5 componentes.

2.2 Estado Inicial

O estado inicial de um problema é baseado dependendo do tipo de estratégias de buscas que estamos utilizando, por exemplo o problema da n rainhas teria como estado inicial o tabuleiro de xadrez vazio.

2.3 Ações

As ações basicamente são o que nosso agente tem de informação baseado no problema para definir um ação a partir de um estado, por exemplo o problema da n rainhas teria como ação eu colocar uma rainha em qualquer posição do meu tabuleiro.

2.4 Modelo de transição

O modelo de transição é um dos componentes mais importantes, pois nele encontra-se a lógica de que a partir de um estado que o agente esteja quais ações eu posso gerar a partir dele, as ações sucessoras geradas nos abre o leque de possibilidades onde podemos

encontrar o nosso objetivo ou se não o próximo passo que teremos que dar.

2.5 Teste Objetivo

O teste objetivo é onde validamos a partir do nossa borda de estados sucessores gerados se encontramos o o estado final e/ou objetivo do nosso problema, se encontrado o estado, percorremos todas as ações geradas até ali para encontrar o caminho ideal para solução do problema.

2.6 Custo do caminho

O custo do caminho, é a quantidade de ações e/ou movimentos que foram gerados definidos por uma expressão dependendo do problema em questão, por exemplo se utilizarmos as estratégias de busca no problema da Torre de Hanoi, onde o estado inicial tendo n números de discos é formulado por $2^n - 1$ onde n é o numero de discos da torre inicial.

3 Analise dos problemas

3.1 Cubo de Rubik

O cubo de Rubik é um problema complexo e de difícil resolução, foi inventado pelo húngaro Ernõ Rubik em 1974, originalmente foi chamado de o "Cubo Mágico"pelo seu inventor, mas o nome foi alterado nos Estados Unidos pela empresa Ideal Toys para "Cubo de Rubik". Esse cubo geralmente é confeccionado em plástico e possui várias versões, sendo a mais comum a de 3x3x3 compostas por 6 faces de 6 cores diferentes, entretanto o cubo possui outras versões 2x2x2, 4x4x4 e 5x5x5 que são menos conhecidas.

Atualmente é possível resolver o cubo de Rubik com apenas 20 movimentos, esse resultado foi obtido utilizando um algoritmo chamado "Algoritmo de Deus", se pararmos para pensar o numero total de combinações do cubo de Rubik é de 43 252 003 274 489 856 000, mesmo se alguém pudesse realizar todas as combinações possíveis a uma velocidade de um movimento por segundo, demoraria 1400 trilhões de anos, supondo que nunca repetisse a mesma combinação.

Nossos testes vão utilizar o algoritmo de busca A * para realizar o problema de busca para o cubo de Rubik, que é o mais indicado devido ao fato de que o que importa é resolver o cubo de Rubik com cada face apenas com uma cor e não estamos se baseando no melhor custo de caminho possível, como existe muitas possibilidades de combinações a busca sem informação não seria valida pois teríamos que passar por cada uma dessas possibilidades até encontrar o estado final, com a busca com informação utilizaremos de conhecimento especifico além do problema para auxiliar no descobrimento do melhor caminho possível, começaremos definindo os nossos 5 componentes de resolução de problema por busca com informação.

Primeiro, o **estado inicial** do nosso problema é o cubo de Rubik, todo bagunçado, conforme imagem abaixo.

Figura 1: Estado Inicial

Podemos pensar nas **ações** que podemos realizar no cubo para gerar os estados, no cubo de Rubik 3x3x3 podemos ter 18 possíveis ações de movimentação.

Figura 2: Ações

O **modelo de transição** do cubo de rubik, utilizando o algoritmo A*, pode ser definido da seguinte forma, primeiro temos que utilizar de informações especificas além do problema por isso definiremos uma função de avaliação para cada estado gerado em uma movimentação, a nossa função de avaliação será definida por $f(n)$ a partir de cada nó, onde.

- $f(n) = g(n) + h(n)$
- $g(n) =$ custo até o momento para alcançar n
- $h(n) =$ estimativa do custo de n até o objetivo

Com a função de avaliação podemos adicionar a nossa borda apenas os estados sucessores que forem admissíveis, onde $h(n) <= h*(n)$, onde $h*(n)$ é o custo verdadeiro de alcançar o estado objetivo a partir de n.

Podemos definir o **estado objetivo** do nosso problema com cada face do cubo de uma cor diferente, isso quer dizer cada um dos 26 cubos de cada face com a mesma cor, sendo a cor de cada face uma diferente da outra.

O **custo de caminho** desse problema pode ser definido pelo numero de ações realizadas do nó filho até o nó pai inicial, tendo com 1 o valor de cada iteração do pai ao nó.

Figura 3: Estado Objetivo

3.2 Missionários e canibais

O problema dos missionários e canibais, conhecido também como problema dos maridos ciumentos, é um clássico quebra-cabeças de travessia de rio, onde encontram-se 3 missionários e 3 canibais na margem esquerda do rio e 1 barco onde o objetivo é atravessar todos os 3 missionários e canibais para a margem direita do rio, mas com as seguintes condições o numero de missionários nas margens nunca pode ser menor que o numero de canibais, pois se não os missionários seriam comidos pelos canibais e o barco não pode atravessar por si só, sem pessoas a bordo.

Utilizando as estratégias de busca, podemos utilizar a busca sem informação ou busca cega como é chamado, para resolver esse problema com o menor numero de viagens possíveis, modelaremos o nosso problema da seguinte forma, a letra C representara os canibais e a letra M os missionários:

- Estado Inicial: será margem_esquerda(C, C, C, M, M, M), barco([ ]) e margem_direita([ ])
- Ações: podemos mover da margem esquerda para a direita 2M ou 2C ou 1M e 1C ou 1M ou 1C
- Modelo de transição: Definimos as regras que de podemos ir da margem esquerda para a direita e o numero de canibais não pode ser maior que o numero de missionários nas margens, após isso geramos os estados sucessores a partir do estado inicial e colocamos na nossa borda os estados vizinhos gerados, será retirado da borda na seguinte ordem o ultimo elemento a entrar será o primeiro a sair.
- Estado objetivo: margem_esquerda([ ]), barco([ ]) e margem_direita(C, C, C, M, M, M)
- Custo do caminho: O custo do caminho é 1 para cada nó pai expandido, tendo em vista que devemos utilizar uma memoria para validar os estados já visitados e qual é o pai e quais os filhos

3.3 Problema das n rainhas

O problema das n rainhas, tem o objetivo de colocar 8 rainhas em um tabuleiro 8x8 (64 posições) de forma que nenhuma rainha ataque outra. Uma rainha ataca outra se esta estiver na diagonal e/ou horizontal e/ou vertical, esse tipo de problema consiste em encontrar o melhor resultado final, mesmo que não seja o ideal, por isso para esse problema iremos utilizar o algoritmo de busca local Hill Climbing. O algoritmo de busca Hill Climbing, tem uma função objetivo que deve encontrar o máximo global e o custo deve ser o mínimo global, a partir de um estado, iniciamos uma pertubação e pegamos o vizinho com o maior máximo global, o algoritmo termina quando encontrar um pico (ou vale) em que nenhum vizinho tem

valor mais alto. Existem tem 3 tipos de Algoritmo Hill Climbing, Steepest-Ascent Hill Climbing (Examina todos os vizinhos e escolhe o melhor), Stochastic hill Climbing (Seçecopma l vozomjps açeat[props e escolhe o melhor) e o Hill Climbing Random Restart (Inicializa o Hill Climbing em diferentes pontos do espaço de busca). Utilizaremos para esse modelagem de problema o Hill Climbing Random Restart tendo em vista que temos 64 posições diferentes que podemos iniciar com uma rainha para obter o resultado final.

- Estado inicial: Inicia com o tabuleiro com uma rainha em uma posição qualquer dentro do tabuleiro
- Ações: Mover as 8 rainhas entre as posições do tabuleiro
- Modelo de transição: Mover a peça em uma posição não ocupada e que não faça relação horizontal, vertical e diagonal com outra peça
- Teste Objetivo: As 8 rainhas estão dispostas no tabuleiro sem que nenhuma esteja atacando outra
- Custo do caminho: 1 para cada movimento

3.4 Sudoku

O jogo de raciocínio lógico Sudoku, é um jogo baseado na colocação lógica de números,que tem por objetivo a colocação de números de 1 a 9 em cada uma das células vazias numa grade de 9x9, constituída por 3x3 subgrades chamadas regiões, o quebra-cabeça contém algumas pistas inicias onde as subgrades já iniciam-se preenchidas deduzindo um possível numero ou caminho a partir daquele ponto. Nosso objetivo é utilizar o algoritmo Hill Climbing para solucionar esse problema por meio de busca.

O algoritmo de busca Hill Climbing, tem uma função objetivo que deve encontrar o máximo global e o custo deve ser o mínimo global, a partir de um estado, iniciamos uma pertubação e pegamos o vizinho com o maior máximo global, o algoritmo termina quando encontrar um pico (ou vale) em que nenhum vizinho tem valor mais alto. Existem tem 3 tipos de Algoritmo Hill Climbing, Steepest-Ascent Hill Climbing (Examina todos os vizinhos e escolhe o melhor), Stochastic hill Climbing (Seleciona k vizinhos aleatórios e escolhe o melhor) e o Hill Climbing Random Restart (Inicializa o Hill Climbing em diferentes pontos do espaço de busca). Utilizaremos para esse modelagem de problema o Stochastic hill Climbing tendo em vista que temos 64 posições diferentes que podemos iniciar com uma rainha para obter o resultado final.

- Estado inicial: Inicia com as sub-grades parcialmente preenchido
- Ações: inserir um numero de 1 a 9 nas posições vazias entre as sub-grades
- Modelo de transição: Inserir o numero em uma posição vazia de uma sub-grade, onde esse numero não se repita na mesma linha, coluna ou sub-grade 3x3.
- Teste Objetivo: Grade 9x9 totalmente preenchida onde nenhum numero esteja repetido na mesma linha, coluna ou sub-grade,
- Custo do caminho: 1 para cada inserção

4 Considerações Finais

Na Inteligência Artificial, os jogos de brinquedo tem uma grande utilização pois podemos modelar qualquer problema utilizando as estratégias definidas por meio de busca, antes tidos como "quebra-cabeças"hoje são resolvidos ou modelados facilmente para uma possível resolução. O que podemos esperar com a evolução da inteligência artificial?

Referências

ESTEVES., B. A. *SOLUÇÕES INTELIGENTES PARA O CUBO DE RUBIK*. 2020. Disponível em: <http://monografias.nrc.ice.ufjf.br/tcc-web/exibePdf?id=2>. Acesso em: outubro. 2020.

NOVRIG., S. J. R. . P. *A Modern Approach.* Prentice Hall, 3rd edition, 2010. Artificial Intelligence.

WIKIPEDIA. *Historia do Cubo de Rubik.* 2020. Cubo de Rubik. Disponível em: <https://pt.wikipedia.org/wiki/Cubo_de_Rubik>. Acesso em: outubro. 2020.

WIKIPEDIA. *Problema dos canibais e missionários*. 2020. Disponível em: <https://pt.wikipedia.org/wiki/Problema_dos_canibais_e_mission\\%C3\\%A1rios\#:~:text=No\\%20problema\\%20dos\\%20canibais\\%20e,n\\%C3\\%BAmero\\%20de\\%20canibais\\%20na\\%20mesma>. Acesso em: outubro. 2020.

WIKIPEDIA. *Sudoku*. 2020. Disponível em: <https://pt.wikipedia.org/wiki/Sudoku>. Acesso em: outubro. 2020.

(ESTEVES., 2020) (WIKIPEDIA, 2020a) (NOVRIG., Prentice Hall, 3rd edition, 2010) (WIKIPEDIA, 2020b) (WIKIPEDIA, 2020c)