Trabalho I

Busca Cega e Busca com Informação

Alisson Jaques
Engenharia da Computação
Universidade do Estado de Minas Gerais-UEMG
Divinópolis,MG
alissonjaquesrmq@gmail.com

Guilherme Gomes Noronha
Engenharia da Computação
Universidade do Estado de Minas Gerais-UEMG
Divinópolis,MG
ggn.noronha@gmail.com

***Abstract*—When faced with problems involving the tree in AI, what is the best method? In this article, we have a notion of how to proceed, based on the search methods that are the method of Blind Search and Search with Information.**

Palavras-chaves—busca cega, busca heurística, árvore, 8-Puzzle

I.INTRODUÇÃO

No âmbito da pesquisa em sistemas inteligentes é comum vermos diferentes métodos de busca em árvore para resolver diferentes tipos de problemas. Em uma análise introdutória somos levados a estudar métodos de busca cega, busca com funções heurísticas, entre outros. Neste trabalho iremos analisar os métodos de busca cega e busca com funções heurísticas, tendo como base uma situação problema.

A situação problema consiste no algoritmo 8-Puzzle, esse problema consiste numa matriz nxn com n algarismos distintos (n = 0,1,.....,n) sendo um dos algarismos 0 (representando o vazio), o objetivo é ordenar de forma crescente os algarismos dada uma configuração inicial aleatória qualquer. A Figura1 mostra um estado possível para o problema 8-Puzzle.

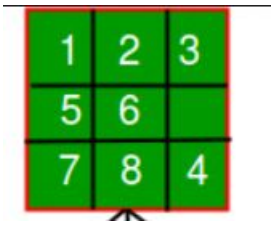

Figura1: Um estado do 8-Puzzle

As trocas são realizadas com o espaço vazio e, dependendo de onde o mesmo se encontra, pode ser necessário ter de escolher até entre quatro diferentes opções. Cada troca gera uma nova matriz, e cada nova matriz gera novas matrizes, sendo assim, podemos representar todos os possíveis estados do 8-puzzle em uma árvore de estados. A Figura2 mostra uma árvore de estados do 8-Puzzle.

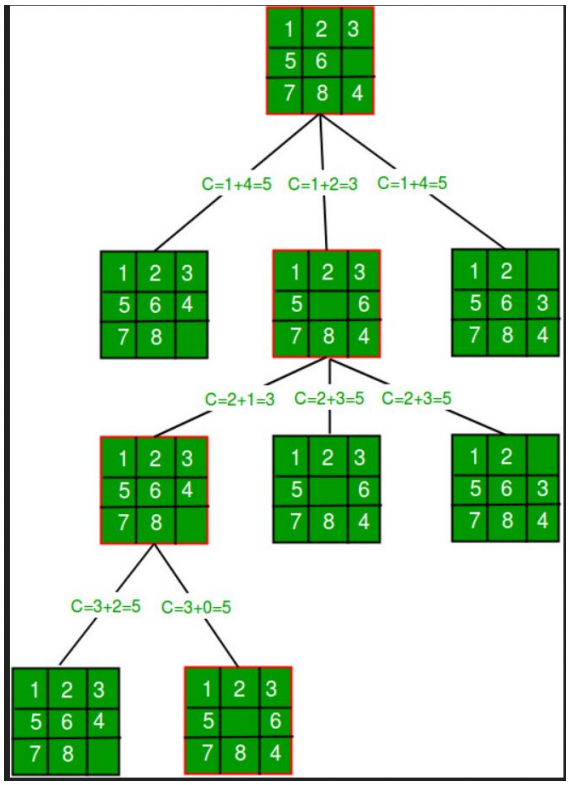

Figura2: Árvore de Estados 8-Puzzle

Tendo como base a situação problema acima, serão feitas comparações entre os métodos de busca cega e busca heurística.

II. ALGORITMOS DE BUSCA

A. Busca Cega

Essa busca se caracteriza por não utilizar uma função inteligente para varrer a árvore, sendo a mesma sendo varrida de forma brusca. Aqui desenvolvemos um algoritmo em C++ que busca cegamente em profundidade, ou seja, uma vez escolhido um estado o mesmo tende a expandir os filhos deste estado. Essa configuração não é ótima e, dependendo da situação, pode nunca encontrar a solução.

Mesmo para instâncias pequenas o algoritmo pode dar diversos loopings desnecessários, pois pode expandir nós que se afastam do estado objetivo. A ideia por trás de sua implementação está na utilização de uma árvore recursiva, manipulação de uma matriz de índices e utilização de uma matriz de estados, a descrição completa pode ser vista no seguinte repositório: (https://github.com/engenharia-da-computacao/trabalho1-si-2019/tree/master), devidamente comentada e indentada.

B. Busca com Informação – Busca Gulosa

.

Essa busca utiliza uma função gulosa que avalia qual o melhor escolha de acordo com o estado atual. A ideia por trás da criação do algoritmo consiste na criação de uma função que utiliza uma matriz de índices para estimar qual a menor distância da peça atual à sua posição ideal. Dessa forma, o algoritmo escolhe uma peça para trocar com o vazio se a mesma ter a menor distâncias entre as demais (métrica gulosa), na vizinhança do vazio. Esse tipo de busca tende a ser muito prático, mas não é ótimo. Para o problema considerado veremos que esse método se mostrou bastante eficaz. A descrição completa, comentada e indentada, pode ser consultada no mesmo repositório, passado anteriormente.

III. COMPARANDO OS ALGORITMOS ATRAVÉS DE MÉTRICAS DE DESEMPENHO

Executamos os métodos simultaneamente para 5 diferentes instâncias, o resultado pode ser verificado na tabela C.

C. Tabela comparando os desempenhos dos dois métodos utilizados

	Instância	Número de Movimentos	Tempo Gasto
Busca Cega	1	1039	53.7004 ms
Guloso	1	139	7.32335 ms
	Instância	Número de Movimentos	Tempo Gasto
Busca Cega	2	942	39.9456 ms
Guloso	2	454	38.6063 ms
	Instância	Número de Movimentos	Tempo Gasto
Busca Cega	3	942	44.3554 ms
Guloso	3	453	7.32335 ms
	Instância	Número de Movimentos	Tempo Gasto
Busca Cega	4	1006	53.723 ms
Guloso	4	452	35.0099 ms
	Instância	Número de Movimentos	Tempo Gasto
Busca Cega	5	1038	53.836 ms
Guloso	5	144	6.08596 ms

Tabela C: Resultado da execução dos métodos para diferentes instâncias

Note que para cada instância a configuração do estado da matriz inicial era a mesma, assim podemos fazer comparações relevantes entre os métodos empregados. Fica claro que ao analisar a Tabela C o algoritmo mais eficiente foi o algoritmo guloso, poupando muito mais tempo e processamento. Neste caso a estratégia gulosa tende a expandir o nó mais próximo do objetivo, enquanto o outro algoritmo busca exaustivamente pelo nó objetivo de forma bruta. Há de se notar que o algoritmo cego por princípio sempre deve encontrar uma solução, desde que a mesma exista, mas para uma análise técnica o algoritmo guloso é o mais indicado, pois mesmo com instâncias grandes a taxa de sucesso dele não deixa a desejar.

REFERÊNCIAS

[1] JUNIOR, Nelson Florêncio; GUIMARÃES, Frederico Gadelha. Problema 8-Puzzle: Análise da solução usando Backtracking e Algoritmos Genéticos.

[2] GASCHNIG, John. A problem similarity approach to devising heuristics: First results. In: **Readings in Artificial Intelligence**. Morgan Kaufmann, 1981. p. 23-29.

[3] GEFFNER, P. Haslum H.; HASLUM, P. Admissible heuristics for optimal planning. In: **Proceedings of the 5th Internat. Conf. of AI Planning Systems (AIPS 2000)**. 2000. p. 140-149.

[4] VON ZUBEN, EA072–Prof Fernando J. Estruturas e Estratégias de Busca.

[5] 8 Puzzle Problem using Branch and Bound. Disponível em:https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/[Acesso:13 Jun-2019]