**Implementação do Jogo dos 15**

FCUP

Inteligência Artificial (CC2006) 2018/2019

Trabalho de Grupo AK

Eduardo Morgado – 201706894 – MIERSI

Simão Cardoso – 201604595 – MIERSI

Sónia Rocha – 201704679 – MIERSI

1. **Introdução:**
   1. **Um problema de Pesquisa:**

Podemos considerar um problema de pesquisa como sendo uma sequência de ações que permitem a chegada a um estado objetivo/final a partir de um estado inicial **[3]** . Cada uma destas ações irá alterar um estado, o conjunto de todos os estados possíveis e os operadores que relacionam os estados entre si formam o espaço de pesquisa **[3,4]** .

Sendo assim um problema de pesquisa corretamente formulado pode ser descrito por um estado inicial do problema (ponto de partida), um conjunto de operadores que levam à criação de novos estados a partir de um dado estado, um espaço de estados acessíveis a partir do inicial por uma sequência qualquer de ações (aplicações de operadores), um caminho (uma sequência de ações) associado a um custo, dependendo este do custo individual de cada ação e por um teste de verificação da chegada ao estado objetivo/final **[1,3]** .

* 1. **Métodos de Busca:**

De entre os vários métodos de pesquisa, os mais usados entram nas categorias de pesquisa não informada/força bruta e pesquisa informada/pesquisa com heurística[[1]](#footnote-1).

* + 1. **Pesquisa Não Informada**

Uma pesquisa não informada é uma pesquisa geral, mais fraca, uma vez que não pressupõe propriedades sobre o problema a tratar todos os algoritmos de pesquisa não informada apresentam complexidade temporal e espacial de e respetivamente (exceto a pesquisa bidirecional, com complexidade temporal e espacial de ) **[2]**, onde d representa a profundidade na árvore de pesquisa e b o fator de ramificação. De entre os métodos de pesquisa não informada encontramos:

* Pesquisa em Largura (BFS) – Análise exaustiva onde, a partir de um nó origem/raiz, todos os nós são gerados por ordem crescente de distância para a raiz **[2]**. O primeiro caminho para o nó objetivo será o de menor comprimento, normalmente, a sua implementação envolve a utilização de uma *queue*.
* Pesquisa em Profundidade (DFS) – A partir de um nó raiz, apenas explora o próximo filho depois de acabar a exploração do primeiro, ou seja, explora um caminho até encontrar o nó objetivo ou ser forçado a regressar, normalmente, pode ser implementado por métodos iterativos, utilizando *stacks*, ou por métodos recursivos, utilizando a *stack* do sistema.
* Pesquisa Limitada em Profundidade (LDFS) – Método suportado por DFS, no entanto, é estipulado *a priori* um limite para a profundidade do caminho a ser percorrido.
* Pesquisa em Profundidade Limitada (IDFS) – Combina vantagens de BFS com DFS, é suportada por uma LDFS onde vamos aumentando o limite fixo em LDFS, garantindo assim o teste de todos os possíveis limites de profundidade, reduzindo o espaço utilizado.
  + 1. **Pesquisa Informada**

Uma pesquisa informada recorre a funções heurísticas para estimar o quão próxima da solução um nó está, essa estimativa será utilizada para conduzi melhor a pesquisa, tornando-a muito mais eficiente que uma pesquisa não informada.

Este método utiliza uma função f(n) para avaliar um nó, retornando um valor que pode servir como incentivador para a escolha do nó em questão. De entre os vários métodos de pesquisa informada encontramos:

* Pesquisa A\* -
* Pesquisa *Greedy* –

1. **Estratégias de Procura:**

Nesta secção iremos ver com mais pormenor os métodos de pesquisa que foram implementados no trabalho prático.

* 1. **Pesquisa Não Informada:**

Já vimos anteriormente que, a pesquisa não informada é uma pesquisa mais exaustiva, como tal, menos eficiente quando comparada com métodos informados.

**2.1.1. Pesquisa em Largura (BFS)**

Este método é inicializado na raiz da árvore de pesquisa e apenas explora nós de nível seguinte após explorar todos os nós do nível atual. É um método completo (isto se houver solução), tem possibilidade de ser ótimo, caso os operadores para o problema tenham todos o mesmo custo uniforme. Este método garante que a solução é encontrada em número de passos ótimo, no entanto, requer muita memória e tempo, uma vez que, é necessário armazenar todos os nós, apresentando uma complexidade temporal e espacial exponencial, sendo a quantidade de memória disponível para resolver o problema o maior limitador deste método, como tal, não é recomendado ser utilizado para problemas com instâncias muito grandes de estrada.

É usual implementar este método com uma *queue* FIFO, onde, quando visitados um nó, expandimo-lo e adicionamos os seus nós ainda não visitados, marcando-os como visitados, ao final da fila. A *Figura 1* apresenta o pseudocódigo para este método.

**BFS(inicial,final):**

1 Q <- MK\_QUEUE(inicial)

2 Enquanto Q\_IS\_EMPTY() != False fazer

3 no <- DEQUEUE(Q)

4 Se no == final então retorna Sucesso

5 Para cada no\_filho fazer

6 Se no\_filho não visitado então

7 visita\_no\_filho

8 ENQUEUE(no\_filho,Q)

Figura 1-Pseudocódigo BFS

Este método tem como complexidade espacial e temporal , uma vez que, para cada nível d, terá que guardar todos os estados gerados por todas as operações válidas dos nós de profundidade d-1.

**2.1.2. Pesquisa em Profundidade (DFS)**

**7. Referências:**

**[1]** École Polytechique Montréal 6 de dezembro de 2000, Michel Gagnon, consultado pela última vez a 4 de março de 2019, [<http://www.professeurs.polymtl.ca/michel.gagnon/Disciplinas/Bac/IA/ResolProb/resproblema.html>](%3chttp://www.professeurs.polymtl.ca/michel.gagnon/Disciplinas/Bac/IA/ResolProb/resproblema.html%3e)

**[2]** Stanford University Computer Science n.d., *Search Methods*, Patrick Doyle, consultado pela última vez a 4 de março de 2019, [<https://users.cs.duke.edu/~brd/Teaching/Previous/AI/Lectures/Summaries/search.html#Game Trees / Adversarial Search>](%3chttps://users.cs.duke.edu/~brd/Teaching/Previous/AI/Lectures/Summaries/search.html#Game Trees / Adversarial Search>)

**[3]** University of Cambridge Department of Engineering Multimedia Group n.d.,*Problem Solving by Search*, Rod Goodyer, consultado pela última vez a 4 de março de 2019, [<http://www-g.eng.cam.ac.uk/mmg/teaching/artificialintelligence/nonflash/problemframenf.htm>](http://www-g.eng.cam.ac.uk/mmg/teaching/artificialintelligence/nonflash/problemframenf.htm)

**[4]** University of Cambridge Department of Engineering Multimedia Group n.d., *What is Search?*, Rod Goodyer, consultado pela última vez a 4 de março de 2019, [<http://www-g.eng.cam.ac.uk/mmg/teaching/artificialintelligence/nonflash/problemframenf.htm>](http://www-g.eng.cam.ac.uk/mmg/teaching/artificialintelligence/nonflash/problemframenf.htm)

1. Uma heurística são regras com possibilidade de solucionar um problema, no entanto, não garantem a existência de uma solução. A partir do conhecimento de propriedades do problema, uma heurística facilita a pesquisa da solução **[2]** . [↑](#footnote-ref-1)