Resolução do jogo Cohesion utilizando Métodos de

Pesquisa em Linguagem Python (Tema 4/ Grupo 33)

João Alves (201605236)  
Faculdade de Engenharia da Universidade do Porto

Porto, Portugal

[up201605236@fe.up.pt](mailto:up201605236@fe.up.pt)

Amadeu Pereira (201605646)  
Faculdade de Engenharia da Universidade do Porto

Porto, Portugal

[up201605646@fe.up.pt](mailto:up201605646@fe.up.pt)

Nuno Lopes (201605337)  
Faculdade de Engenharia da Universidade do Porto

Porto, Portugal

[up201605337@fe.up.pt](mailto:up201605337@fe.up.pt)

*TODO: Resumo*—Este trabalho tem como objetivo a aplicação de vários métodos de pesquisa na resolução de vários níveis de um jogo (cohesion). A aplicação permite dois modos de jogo (jogador / computador) em que, para o computador, pode ser escolhido o método a ser utilizado para a resolução do puzzle.

O código fonte é escrito em python, o que ajudou no desenvolvimento do trabalho. A parte do computador (Inteligência Artificial) foi testada com todos os algoritmos de pesquisa desenvolvidos, variando a dificuldade dos níveis (easy-medium-hard) obtendo quase sempre uma resposta positiva. À medida que é aplicado um método é feita uma análise aos seus custos que é representada em modo texto pela aplicação. De um modo geral os resultados pretendidos aproximam-se aos resultados esperados confirmando a teoria estudada.

***Keywords: Inteligência Artificial, Pesquisa, Algoritmo A\*, Cohesion, Pathfinding, Procura em Profundidade, Procura em Largura, Custo Uniform, Aprofundamento Iterativo, Pesquisa Gulosa***

# Introdução

TODO: Alguns parágrafos motivando e introduzindo o tema e os objetivos do documento e descrevendo a estrutura do artigo.

# Descrição do Problema

O jogo Cohesion é um puzzle baseado no jogo “15 Puzzle” onde os quadrados têm várias cores. As peças podem ser jogadas para qualquer lado desde que não haja nenhuma peça de outra cor que impeça.

Se dois quadrados/peças da mesma cor se tocam eles unem-se num só permanentemente, originando assim uma peça nova.

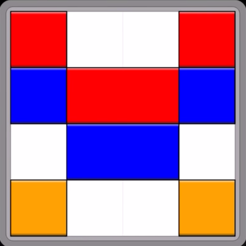


Ilustração 1 - Exemplo de um nível do puzzle.

O objetivo deste puzzle é juntar todas as peças da mesma cor.

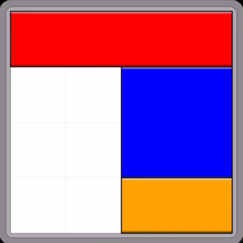


Ilustração 2 - Exemplo de um estado objetivo do puzzle.

# Formulação do Problema

Representação do Estado: Matriz 4x4 em que cada elemento pertence a {0, ..., N}, sendo N o número total de cores distintas existentes no jogo. Quando um elemento é 0 significa que não existe nenhuma peça (cor) nesta posição.

Estado Inicial: O estado inicial depende do nível em questão. Consiste numa distribuição de várias cores pela matriz, garantindo que as peças da mesma cor não se encontram todas juntas.

Estado Objetivo: As peças da mesma cor estão todas adjacentes umas às outras.

Operadores:

Quando 2 peças da mesma cor se tocam formam um bloco permanente, tendo este conjunto de peças da mesma cor de se mover em conjunto.

* **esquerda**(pos):

Pré-Condição: todas as posições à esquerda do bloco da peça selecionada(pos) tem que ser uma posição vazia.

Efeito: o bloco anda uma posição para a esquerda.

Custo: 1.

* **direita**(pos):

Pré-Condição: todas as posições à direita do bloco da peça selecionada(pos) tem que ser uma posição vazia.

Efeito: o bloco anda uma posição para a direita.

Custo: 1.

* **cima**(pos):

Pré-Condição: todas as posições em cima do bloco da peça selecionada(pos) tem que ser uma posição vazia.

Efeito: o bloco anda uma posição para cima.

Custo: 1.

* **baixo**(pos):

Pré-Condição: todas as posições em baixo do bloco da peça selecionada(pos) tem que ser uma posição vazia.

Efeito: o bloco anda uma posição para baixo.

Custo: 1.

Custo da solução: Cada movimento custa 1, logo o custo da solução é o número total de movimentos.

# Trabalho Relacionado

Não conseguimos encontrar nenhum código-fonte do jogo Cohesion, logo como tal decidimos basearmo-nos em implementações do jogo “15 Puzzle” uma vez que tem certas semelhanças ao nosso puzzle. Sendo assim encontramos a implementação de uma pessoa chamada Milan Pecov [2] onde implementa algoritmo A\* e pesquisa em largura para o jogo “15 Puzzle”.

Também para a implementação deste jogo vamos utilizar o código disponível no link [1] pois é possível encontrar a implementação do jogo em diversas linguagens de programação.

TODO: ACRESCENTAR MAIS CENAS

# Implementação do Jogo

TODO: Descrevendo o projeto e implementação, na linguagem selecionada, do jogo incluindo a forma de representação do estado do tabuleiro, operadores (verificação do cumprimento das regras do jogo) aplicáveis com determinadas pré-condições e que têm efeitos sobre o estado do jogo e um dado custo, teste objetivo (determinação do final do jogo). Entre outras devem ser implementadas funções: ler nível de ficheiro (lendo um dado nível/estado de um ficheiro de texto), visualizar em modo de texto/gráfico um dado estado, validar uma dada jogada/operador (tendo em conta as suas pré-condições), executar uma dada jogada/operador, num dado tabuleiro, tendo em conta os seus efeitos e gerando o respetivo estado sucessor, listar todas as jogadas/operadores disponíveis num dado tabuleiro, avaliar um dado estado (tendo em conta a sua “proximidade” à solução final), testar se um dado estado é solução (teste objetivo). Os métodos de pesquisa para cálculo das jogadas a realizar que permitam ao computador jogar sozinho e resolver os puzzles devem ser descritos na secção seguinte assim como o método geral para os chamar e resolver o puzzle (utilizando um dado método selecionado de entre os disponíveis).

# Algoritmos de Pesquisa

TODO: Descrevendo os vários algoritmos de pesquisa utilizados e a sua implementação de modo a calcular a próxima jogada do PC ou retornar a solução final (conjunto de operações para transformar o estado inicial no estado objetivo). Devem ser implementados algoritmos para cálculo da solução utilizando pesquisa em largura, pesquisa em profundidade (se aplicável), aprofundamento progressivo, custo uniforme (se aplicável), pesquisa gulosa e Algoritmo A\* (estes último método utilizando várias heurísticas).

# Experiências e Resultados

TODO: Descrevendo as experiências realizadas com os vários algoritmos para resolver diversos puzzles e os resultados obtidos a nível de tempo e custo da solução obtida em cada nível, por cada um dos métodos experimentados. Devem ser incluídas tabelas comparativas dos resultados obtidos na aplicação dos vários métodos aos vários puzzles (níveis do jogo) e discutidos os resultados.

# Conclusões e Perspetivas de Desenvolvimento

TODO: Sumário do trabalho realizado e conclusões que retira deste projeto. Análise crítica dos resultados obtidos em comparação com os resultados teóricos que seriam esperados. Trabalho futuro, ou seja, formas de melhorar o trabalho desenvolvido.

Em primeiro lugar, é possível concordar que o conhecimento sobre métodos de pesquisa foi consideravelmente aprofundado, mostrando-se assim o grupo satisfeito com o trabalho desenvolvido. Visto ser uma área de estudo cada vez mais desenvolvida e devido ao seu aspeto futurista, a IA(Inteligência Artificial) suscitou em nós um grande interesse pela mesma, o que também contribuiu para a realização do trabalho/estudo feito.

Vários testes foram realizados de forma a recolher informação sobre o comportamento dos métodos de pesquisa utilizados. Estes foram feitos fazendo variar o nível de dificuldade do puzzle e a heuristica (se fosse o caso) para cada um dos algoritmos. A análise efetuada permitiu estudar os aspetos que consideramos mais importantes como o tempo de execução, o espaço ocupado e o custo(movimentos executados).

Passando à individualização dos algoritmos foi possível concluir o seguinte :

Como seria expectável, a pesquisa em largura chega à solução óptima mas demora mais tempo, isto pode ser verificado nos resultados apresentados.

A\* é o algoritmo privilegiado dado que se apresenta como o melhor algoritmo, efetivamente é o que acontece quando utilizamos a terceira heurística criada.

O algoritmo de pesquisa em profundidade apesar de se apresentar como um método rápido não é eficaz e nos testes feitos quase nunca chega à melhor solução.

Dado que o custo de cada movimento é sempre igual, e apesar de não fazer muito sentido no contexto do nosso problema, foram igualmente feitos testes para o método de custo uniforme.

Com a pesquisa gulosa nem sempre se chega à melhor solução mas podemos considerar que o tempo de execução é bastante rápido, como é esperado este algoritmo procura explorar sempre o nó com melhor heurística.

Quanto ao "Aprofundamento Iterativo" é possível verificar que corresponde ao esperado pois os valores de tempo confirmam que é bastante mais rápido que o bfs e que apresenta sempre a melhor solução possível.

De um modo geral acreditamos que cumprimos com os requerimentos do trabalho, as experiências efetuadas e os resultados obtidos refletem um estado positivo do trabalho desenvolvido.

##### Referências Bibliográficas

1. Rosetta Code, “15 Puzzle Game", last updated March 2019, [online], available at: <https://rosettacode.org/wiki/15_Puzzle_Game>, consulted on March 2019.
2. Milan Pecov, “15-puzzle", 2013, [online], available at: <https://github.com/MilanPecov/15-Puzzle-Solvers>, consulted on March 2019.