Resolução do Cohesion utilizando Métodos de Pesquisa em Linguagem Python (Tema 4/Grupo 7)

João Miguel Vaz Tello da Gama Amaral (up201708805)
Departamento de Engenharia Informática
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
up201708805@fe.up.pt

João Nuno Rodrigues Ferreira (201605330)
Departamento de Engenharia Informática
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
up201605330@fe.up.pt

Tiago Pinho Cardoso (201605762)
Departamento de Engenharia Informática
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
201605762@fe.up.pt

***Resumo*— Pretende-se neste trabalho implementar um jogo do tipo solitário para um jogador e resolver diferentes versões desse jogo, utilizando métodos de pesquisa. Para esta entrega intercalar será descrito neste artigo o problema, a formulação do mesmo, trabalho relacionado e conclusões e perspetivas de desenvolvimento.**

Keywords—Inteligência Artificial, Pesquisa, Pesquisa em Largura, Pesquisa em Profundidade, Pesquisa de Aprofundamento Progressivo, Pesquisa de Custo Uniforme, Pesquisa Gulosa, Algoritmo A*

I. Introdução

Este trabalho consiste na implementação de Métodos de Pesquisa no âmbito de resolver diferentes versões do 7 jogo escolhido, do tipo solitário. Para esse efeito, será necessário utilizar os métodos adequados, sejam pesquisa em largura, pesquisa em profundidade, aprofundamento progressivo, pesquisa de custo uniforme, pesquisa gulosa e algoritmo A*.

Um jogo do tipo solitário caracteriza-se pelo tipo de tabuleiro, peças, regras de movimentação, condições de terminação do jogo com derrota ou vitória e com a respetiva pontuação.

Pretende-se desenvolver então uma aplicação para jogar estes jogos, neste caso, Cohesion. A aplicação deverá ter uma visualização em modo de texto ou gráfico para mostrar a evolução do tabuleiro e realizar a comunicação com o utilizador/jogador.

Para esta entrega intercalar serão incluídos no artigo a descrição do problema, formulação do problema, trabalho relacionado e conclusões e perspetivas de desenvolvimento. A descrição do problema consiste numa sucinta descrição do solitário e as suas regras. A formulação do problema irá descrevê-lo como um problema de pesquisa, descrevendo a representação do estado, teste objetivo, operadores, pré-condições, efeito e custo.

II. Descrição do Problema

Cohesion é um puzzle baseado no clássico “15 puzzle”, também conhecido por “slide the square” ou “slide puzzle”, porém o Cohesion difere de uma maneira importante.

Os quadrados do jogo Cohesion têm várias cores. Se dois quadrados da mesma cor se tocam, eles unem-se formando uma só peça. A ligação é permanente e a nova forma criada aumenta o desafio de Cohesion.

O puzzle é resolvido assim que todos os quadrados de cada cor forem combinados em uma só peça. O jogo está organizado em 240 puzzles categorizados em três níveis de dificuldade. Na imagem seguinte podemos ver uma possível resolução de um dos níveis.

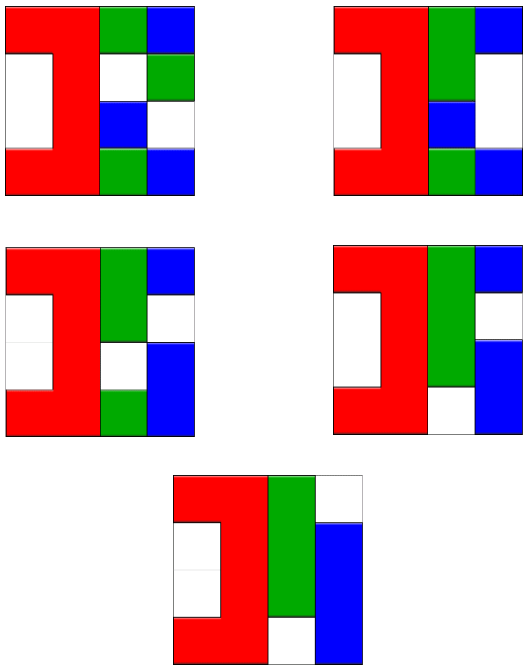

Fig. 1 – Possível resolução de um nível. A ordem das imagens é da esquerda para a direita e de cima para baixo.

III. Formulação do Problema

Representação do estado do jogo: O jogo irá ser representado por uma matriz de tamanho 4x4. Nesta matriz, todos os elementos irão ser representados por um valor que se encontra no intervalo entre 0 e N, sendo que 0 representa uma posição sem qualquer elemento e todos os outros valores até N representam uma cor diferente das outras.

Estado Inicial: O estado inicial irá depender do nível escolhido. A única restrição neste campo, é que todas as peças da mesma cor não podem estar juntas visto que isso iria corresponder a um estado final.

Operadores: Há quatro ações possíveis: esquerda, direita, cima ou baixo. Cada uma destas ações ocorre numa peça ou num conjunto de peças adjacentes da mesma cor. Para que a açáo possa ser realizada, tanto a peça como o conjunto não podem estar a ser bloqueadas por outras peças.

Teste objetivo: O teste objetivo irá verificar se o estado atual do jogo é um estado final, sendo que este é um estado no qual todas as peças da mesma cor se encontram juntas.

Custo da solução: Neste caso, o custo de cada passo será um. Assim sendo, o custo da solução que resolve o problema é o número de passos para chegar à tal solução.

IV. Trabalho Relacionados

No caso do jogo escolhido pelo nosso grupo, Cohesion, cuja página no Google Play é [1], não nos foi possível encontrar código fonte do jogo em si.

Por outro lado, encontrámos implementações de algoritmos que nos serão úteis para a realização do projeto no site [2], visto que a implementação destes é já feita em Python, linguagem na qual iremos trabalhar.

V. Implementação do Jogo

Este projeto foi realizado na linguagem Python. Para implementar o jogo foi necessário estabelecer uma forma de representação do estado do jogo, bem como operadores que afetam o estado do jogo ou que o avaliam e funções auxiliares como a que lê os níveis de um ficheiro de texto.

Inicialmente, são chamadas as funções *parseFile*, *chooseAlg* e *chooseLevel* que, respetivamente, leem de um ficheiro de texto todos os níveis possíveis e fornecem ao utilizador a possibilidade de escolher o nível e o algoritmo que o irá resolver. De seguida, é criada uma instância da classe Game, onde se começa por criar um tabuleiro - variável *board* (array de objetos da classe Piece) - com a função *createPieceBoard* e um array de objetos da class Block – variável *blocks* (que é um conjunto de Pieces que se encontrem juntas) - com a função *createBlocks*.

Tendo por base as variáveis referidas no parágrafo anterior, são inicializados os métodos de pesquisa que serão descritos na secção seguinte.

Por fim, é necessário também referir a implementação das funções que são essenciais para a realização dos movimentos que são elas *ableToMove*, *moveBlock*, *clean_blocks*, para listar todas as jogadas disponíveis, *gameChilds*, e para testar se um estado é solução, *checkWin*. A primeira função recebe como parâmetros um game, um bloco e uma direção e retorna 0 ou 1, dependendo se o movimento é possível ou não. O predicado *moveBlock* é chamado com parâmetros iguais ao anterior, caso seja exequível a realização do movimento. Já a *clean_blocks* é chamada após a *moveBlocks* e realiza a "limpeza" dos blocos, ou seja, peças da mesma cor que se encontrem juntas, mas que só após a última jogada, passam a fazer parte do mesmo block. Em relação à *gameChilds*, esta recebe um estado e retorna todos os estados filho, sendo estes aqueles que sucedem ao estado recebido como parâmetro. Ultimamente, a função *checkWin* retorna se um estado é solução, fazendo a verificação se apenas existe, para cada cor, um bloco.

VI. Algoritmos de Pesquisa

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* (este último método utilizando várias heurísticas).

VII. Experiências e Resultados

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.

VIII. Conclusões e Perspetivas de Desenvolvimento

A realização deste relatório contribuiu para uma melhor compreensão do projeto que iremos realizar. Desta forma, temos uma ideia mais concisa daquilo que teremos de fazer, como o fazer e como dividir o trabalho para que este seja realizado da forma mais clara possível.

Até este momento, já foram desenvolvidas funções que irão ser úteis para a implementação da inteligência artificial no projeto, tais como predicados para movimentar peças e a condição de terminação.

Assim sendo, para o futuro resta-nos a realização dos algoritmos pesquisa que irão resolver os vários níveis do jogo.

Referências Bibliográficas

[1] https://play.google.com/store/apps/details?id=com.NeatWits.CohesionFree.

[2] https://www.redblobgames.com/pathfinding/a-star/implementation.html.