Resolução de Bloxorz utilizando Métodos de Pesquisa em Linguagem C++ (Tema 5/ Grupo 17)

1º Afonso Azevedo (up201603523)
MIEIC
FEUP
Porto, Portugal
up201603523@fe.up.pt

2º Gonçalo Santos (up201603265)
MIEIC
FEUP
Porto, Portugal
up201603265@fe.up.pt

3ª Susana Lima (up201603634)
MIEIC
FEUP
Porto, Portugal
up201603634@fe.up.pt

***Resumo*—Formulação do puzzle "*Bloxorz: Roll the Block*" como um problema de pesquisa, utilizando diversas Heurísticas e métodos de pesquisa, como aprofundamento progressivo, pesquisa gulosa, A*, entre outros.**

Keywords—Inteligência Artificial, Pesquisa, Algoritmo A*, Bloxorz

I. Introdução

Pretende-se com este trabalho explorar os diferentes métodos de pesquisa, associados a diversas heurísticas, na resolução de um problema de pesquisa. O objetivo deste relatório é comparar e analisar as diferentes estratégias e concluir qual a mais adequada a aplicar na construção de um agente racional, capaz de jogar *Bloxorz*.

Neste artigo estão presentes a descrição do puzzle, a sua representação virtual, referências de outros trabalhos utilizados e as conclusões sobre o projeto e os resultados obtidos.

II. Descrição do Problema

Bloxorz é um puzzle cujo objetivo incide no transporte de um bloco através de rolamentos sucessivos do mesmo, desde uma casa inicial até uma casa de destino. A primeira versão deste jogo, com 33 níveis, foi publicada no site *MiniClip* em 2007 [1]. O tabuleiro encontra-se dividido em inúmeras casas (*2 na figura 1*), sendo o seu tamanho e composição (posicionamento da casa de início (*1*), de destino (*3*), de casas interditas (*4*) e de bónus (não visíveis na imagem)) variáveis entre diferentes níveis.

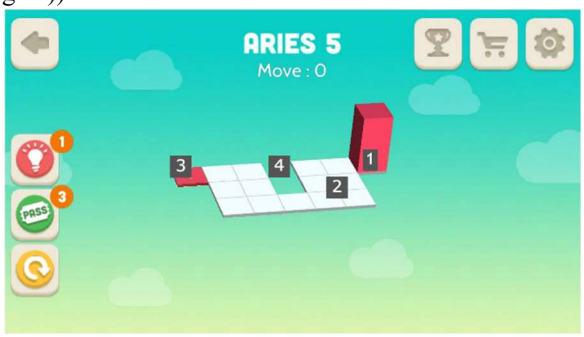

Figura 1-Posição de pé [4]

O bloco pode se movimentar para cima, baixo, esquerda e direita e pode estar posicionado tanto de pé, ocupando apenas uma casa do tabuleiro de jogo, como deitado, ocupando duas casas do tabuleiro. Inicialmente o bloco encontra-se de pé sendo que o puzzle apenas é completado quando o bloco é posicionado de pé na casa de destino. Estão presentes, também, alguns bónus que permitem ao bloco ser diretamente transportado de uma casa para outra (teletransporte) ou desbloquear/bloquear casas adicionais.

Figura 2 - Posição deitada [4]

III. Formulação do Problema

Uma possível formulação do problema como um problema de pesquisa passa por representar o estado através de uma estrutura de dados que contenha o mapa do nível a cada instante, a posição atual do bloco, a posição do destino e o custo atual. Para evitar explorar nós repetidos, seria também guardada uma lista das posições anteriores e os botões ativos em cada turno.

A posição do bloco, tendo em conta que este pode estar de pé (ocupando uma casa) ou deitado (ocupando duas casas), pode ser representada, respetivamente, por um par de coordenadas (x, y) ou por dois pares de coordenadas ((x1, y1), (x2, y2)), correspondentes às coordenadas das casas ocupadas.

Deste modo, e considerando que inicialmente o bloco se encontra de pé na casa de coordenadas (xi, yi), o estado inicial seria representado por (x=xi, y=yi).

Uma vez que o jogo termina com o bloco posicionado de pé na casa de destino, representada pelas coordenadas (xf, yf), o estado final seria (x=xf, y=yf).

Os operadores a considerar nesta representação estão descritos na tabela do Anexo 1.

IV. Trabalho Relacionado

Como forma de contextualizar e inicializar o desenvolvimento do trabalho, foram pesquisados e analisados dois projetos cujo foco incide na resolução do puzzle, B*loxorz* (pelo utilizador grahambarrgraham no GitHub) [2] e *bloxorz_solver* (pelo utilizador *alppboz* no GitHub) [3]. Ambos implementam o algoritmo A*, no entanto o segundo implementa, também, o algoritmo Uniform Cost Search. O primeiro, implementado em Java, permitiu uma análise mais aprofundada da implementação do algoritmo, enquanto que o segundo, implementado em Python, forneceu informação adicional relativamente às funções de heurística a utilizar.

No entanto, é de referir que todo o código implementado no trabalho foi desenvolvido pelos alunos, sendo os projetos externos analisados usados de forma unicamente informativa.

V. Conclusões e Perspetivas de Desenvolvimento

O trabalho desenvolvido nesta etapa consistiu na elaboração do jogo como um problema de pesquisa e na representação dos operadores e da lógica de jogo de uma forma vantajosa para a resolução do problema.

Em suma, de acordo com o documento e o código estudados, é previsto que os resultados do agente implementado encontrarão soluções com diferentes graus de eficiência. Os diversos métodos utilizados pelo agente irão influenciar a performance na operação da descoberta da solução do nível, é antecipado que os melhores resultados sejam conseguidos pelo método A*. Várias heurísticas serão elaboradas para otimizar o processo de procura de soluções e o seu impacto será analisado de acordo com os níveis do jogo e dos métodos utilizados.

Referências Bibliográficas

[1] FANDOM Games Community, "Bloxorz | MiniClip Wiki," FANDOM, 24 Novembro 2012. [Online]. Available: https://miniclip.fandom.com/wiki/Bloxorz. [Acedido em 3 Março 2019].

[2] grahambarrgraham, "bloxorz," GitHub, [Online]. Available: https://github.com/grahambarrgraham/bloxorz. [Acedido em 18 Março 2019].

[3] alppboz, "bloxorz_solver," GitHub, [Online]. Available: https://github.com/alppboz/bloxorz_solver. [Acedido em 19 Março 2019].

[4] BitMango, "BitMango," Google Play Store, [Online]. Available: https://play.google.com/store/apps/details?id=com.bitmango.go.bloxorzpuzzle. [Acedido em 18 Março 2019].

Anexo 1

Considerem-se as seguintes definições:

sideways: y1 == y2 && x1 != x2
forwards: y1 != y2 && x1 == x2
standing: y1 == y2 && x1 == x2
max(a, b): máximo entre a e b
min(a, b): mínimo entre a e b
cols: número de colunas do tabuleiro/plataforma de jogo
rows: número de linhas do tabuleiro/plataforma de jogo
emptyCell: casa vazia (válida e sem bónus) para a qual o bloco se irá movimentar
transportCell(a): casa para a qual o bloco se irá movimentar que desencadeia o teletransporte para a casa transportCell(a') correspondente de coordenadas (x=xtc(a), y=ytc(a)), se a=0 a'=1, se a=1 a'=0,
switchCell(a): casa para a qual o bloco se irá movimentar que desbloqueia (a=1) ou bloqueia (a=0) casas adicionais

Nome	**Pré-condições**	**Efeitos**	**Custo da operação**
Movimento para cima	IF sideways && y1 > 0 && emptyCell	x1=x1; y1=y1-1; x2=x2; y2=y2-1	1
	IF forwards && min(y1, y2) > 0 && emptyCell	x=x1; y=min(y1, y2)	1
	IF forwards && min(y1, y2) > 0 && transportCell(0)	x=xtc(1); y= ytc(1)	1
	IF forwards && min(y1, y2) > 0 && transportCell(1)	x=xtc(0); y= ytc(0)	1
	IF forwards && min(y1, y2) > 0 && switchCell(0)	x=x1; y=min(y1, y2); casas adicionais bloqueadas	1
	IF forwards && min(y1, y2) > 0 && switchCell(1)	x=x1; y=min(y1, y2); casas adicionais desbloqueadas	1
	IF standing && y1> 1 && emptyCell	x1=x1; y1=y1-2; x2=x2; y2=y2-1	1

Movimento para baixo	IF sideways y1 < rows -1 && emptyCell	x1=x1; y1=y1+1; x2=x2; y2=y2+1	1
	IF forwards && max(y1, y2) < rows - 2 && emptyCell	x=x1; y=max(y1,y2)+1	1
	IF forwards && max(y1, y2) < rows - 2 && transportCell(0)	x=xtc(1); y= ytc(1)	1
	IF forwards && max(y1, y2) < rows - 2 && transportCell(1)	x=xtc(0); y= ytc(0)	1
	IF forwards && max(y1, y2) < rows - 2 && switchCell(0)	x=x1; y=max(y1, y2) +1; casas adicionais bloqueadas	1
	IF forwards && max(y1, y2) < rows - 2 && switchCell(1)	x=x1; y=max(y1, y2) +1; casas adicionais desbloqueadas	1
	IF standing && y1< rows – 2 && emptyCell	x1=x1; y1=y1+2; x2=x2; y2=y2+1	1
Movimento para a esquerda	IF sideways && min(x1,x2) > 0 && emptyCell	x=min(x1, x2)- 1; y=y1	1
	IF sideways && min(x1,x2) > 0 && transportCell(0)	x=xtc(1); y= ytc(1)	1
	IF sideways && min(x1,x2) > 0 && transportCell(1)	x=xtc(0); y= ytc(0)	1
	IF sideways && min(x1,x2) > 0 && switchCell(0)	x=min(x1, x2) - 1; y = y1; casas adicionais bloqueadas	1
	IF sideways && min(x1,x2) > 0 && switchCell(1)	x = min(x1, x2) - 1; y = y1; casas adicionais desbloqueadas	1
	IF forwards && x1 > 0 && emptyCell	x1=x1-1; y1=y1; x2=x2-1; y2=y2	1
	IF standing && x1 > 1 && emptyCell	x1=x1-2; y1=y1; x2=x1-1; y2=y2	1
Movimento para a direita	IF sideways && x1 < cols -1 && emptyCell	x=max(x1,x2)+1 ; y=y1	1
	IF sideways && x1 < cols -1 && transportCell(0)	x=xtc(1); y= ytc(1)	1
	IF sideways && x1 < cols -1 && transportCell(1)	x=xtc(0); y= ytc(0)	1
	IF sideways && x1 < cols -1 && switchCell (0)	x=max(x1,x2)+1 ; y=y1; casas adicionais bloqueadas	1
	IF sideways && x1 < cols -1 && switchCell (1)	x=max(x1,x2)+1 ; y=y1; casas adicionais desbloqueadas	1
	IF forwards && max(x1,x2) < cols – 2 && emptyCell	x1=x1+1; y1=y; x2=x2+1; y2=y2	1
	IF standing && x1 < cols -1 && emptyCell	x1=x1+2; y1=y1; x2=x1+1; y2=y2	1