Unblock Me

Guilherme Silva (201603647)
IART
FEUP
Porto, Portugal
up201603647@fe.up.pt

Miguel Duarte (201606298)
IART
FEUP
Porto, Portugal
up201606298@fe.up.pt

Rui Alves (201606746)
IART
FEUP
Porto, Portugal
up201606746@fe.up.pt

- Update abstract
- Update introduction
- Fix Operators - Move several cells with cost 1 instead of just 1
- Game implementation section
- search algorithms section
- experiments and results section
- update conclusion

***Abstract*—The aim of this paper is to implement and compare Artificial Intelligence algorithms for solving the game *Unblock Me*, in which rectangular pieces block the path of a red special piece. The objective of the game is for the special piece to reach the exit of the level, which is achieved by moving it and the other pieces. The problem will be described and then formulated as a *Search Problem*, together with some discussion on the reasoning behind some of the decisions taken during this process. Several algorithms will be investigated, along with testing them against levels with varying difficulty and branching factors. Both the efficiency in reaching a solution and the cost of the calculated solution will be taken into account in the pursuit for the optimal *Search Algorithm*.**

***Index Terms*—Artificial Intelligence, Search Problems, Path-finding algorithms, Decision-making Heuristics, Graph Algorithms, Depth-First Search, Breadth-First Search, Uniform Cost Search, Iterative Deepening, Greedy Search, A*, Bi-Directional Search**

I. Introduction

The game that is the subject of study in this paper (*Unblock Me*) will be approached with several different *Search Algorithms*, implemented in *Python*, in order to conclude what is the best approach for solving it using *Artificial Intelligence Algorithms*.

Firstly, the game will be described in further detail, ensuring that all non-formal aspects of it are well documented. Secondly, it will be formulated as a *Search Problem*, with the approach taken in doing so being illustrated in increased depth, providing reasoning for the decisions taken. Thirdly, a research on related work on this topic will be presented, giving some insight on the problem from other points of view.

Finally, a small section will cover the conclusions and future work, by expanding on the already developed tasks and elaborating on the planned approach.

II. Problem Description

Unblock Me is a puzzle game that was released in 17-06-2009 by *Kiragames*. It is an adapted version of *Rush Hour*, a puzzle game invented by Nobuyuki Yoshigahara in the 1970s. Each level consists of a 6x6 board, surrounded with walls (except for the puzzle's exit) [1].

Fig. 1. Example of an *Unblock Me* level

The game's objective is to move a special piece to the level's exit, by moving that and the other pieces with the least number of movements possible. Pieces are rectangles with a given orientation (vertical or horizontal) and constant length. Pieces can only move in the direction of their orientation into empty cells (they may not overlap).

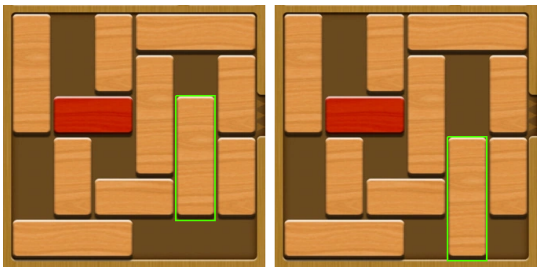

Fig. 2. Piece moving down one cell

Levels are fully surrounded by walls, except for the level's exit door that is alligned with the special piece and by which only the special piece can go through. The level is completed once the piece goes through the exit door.

Fig. 3. Beating a level by going through the exit

Some levels may even contain fixed blocks that can not be moved, representing obstacles.

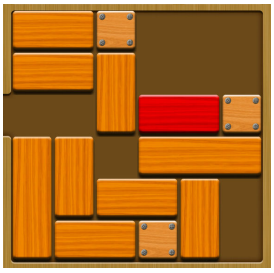

Fig. 4. Example of an *Unblock Me* level containg fixed blocks

III. Problem Formulation

The game's solving process can be formulated as a search problem, in which the goal is to find the sequence of moves (state transitions) that take the special piece to the level's exit door - the problem's goal state.

A. Game State Representation

- List of pieces, where each piece contains the following information:
  - Origin Square (top left piece corner, e.g. (0,0))
  - Length (e.g. 4)
  - Direction (H or V)
- Reference to the special piece
- Matrix of booleans, where True means the cell is empty

This representation of the game state makes the generation of valid movements easier, due to the fact that to determine possible movements of a single piece, the other pieces do not need to be taken into account (assuming that all empty cells are known). Thus, to generate all possible branching options, the list of pieces needs to be iterated over, while checking if there are empty cells adjacent to the piece's extremities (these change with the piece's direction). This representation facilitates the execution of these tasks, making it less costly (important fact due to high number of operations of this kind):

- Using a list of pieces makes it trivial to determine which of them can be moved in a given state;
- Using a matrix (of booleans, that state if a cell is occupied or not) makes it trivial to check if adjacent cells to the pieces extremities are empty;
- Finnaly, using a reference to the special piece allows accessing it in constant time, which will be useful to heuristic related operations and determine if the goal state (subsection III-C) is achieved.

B. Initial State

The initial state depends of the level in question, being represented by a game state as described in subsection III-A.

C. Goal State

The goal state consists of a piece configuration where the special piece reaches the level exit (Figure 3).

D. Operators

- Move piece to the left:
  - Pre-conditions:
    - Piece with horizontal orientation
    - Cell that is adjacent to the piece's left extremity must be empty
  - Results: The pieces position moves one cell to the left
  - Cost: 1 movement
- Move piece to the right:
  - Pre-conditions:
    - Piece with horizontal orientation
    - Cell that is adjacent to the piece's right extremity must be empty
  - Results: The pieces position moves one cell to the right
  - Cost: 1 movement
- Move piece up:
  - Pre-conditions:
    - Piece with vertical orientation
    - Cell that is adjacent to the piece's top extremity must be empty
  - Results: The pieces position moves one cell up
  - Cost: 1 movement
- Move piece down:
  - Pre-conditions:
    - Piece with vertical orientation
    - Cell that is adjacent to the piece's bottom extremity must be empty
  - Results: The pieces position moves one cell down
  - Cost: 1 movement

E. Path cost

The solution's path cost that is to be minimized is equal to the number of movements made (number of state transitions).

IV. Related Work

Unblock Me is also widely known as *Rush Hour*. In June 2018, Michael Fogleman (a Software Engineer at Formlabs) developed an artificial intelligence that solves the puzzle and created a database with over two million and a half different puzzles [2]. His solution is based in the usage of bitboards [3] (data structure that is often used by board games computer systems due to its short space usage and operations efficiency) for the game states representation.

This game is also studied in the Computer Science MSc from the University of Princeton, namely in the course "Computer Science 402 - Artificial Intelligence". In this course, students implement multiple heuristics to solve the problem using the A* algorithm [4]. Rainhard Findling, a researcher in Aalto University in Finland, proposes a solution to this problem statement, by using a heuristic based on a vote system, where each piece that is blocking the way of the special piece to the level's exit votes on how they want to free the way (how many moves they have to perform) for the special piece [5].

Even though that the problem is not complex, the increasing of the board size quickly leads to a combinatorial explosion, being that a 6x6 board has over 27 billion possible states. Thus, it is not efficient to find a solution by exploring all the possible states, being a good heuristic for state exploration mandatory.

V. Game Implementation

Descrevendo o projeto e implementao, na linguagem selecionada, do jogo incluindo a forma de representao do estado do tabuleiro, operadores (verificao do cumprimento das regras do jogo) aplicveis com determinadas pr-condies e que tm efeitos sobre o estado do jogo e um dado custo, teste objetivo (determinao do final do jogo). Entre outras devem ser implementadas funes: ler nvel de ficheiro (lendo um dado nvel/estado de um ficheiro de texto), visualizar em modo de texto/grfico um dado estado, validar uma dada jogada/operador (tendo em conta as suas pr-condies), 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 disponveis num dado tabuleiro, avaliar um dado estado (tendo em conta a sua proximidade soluo final), testar se um dado estado soluo (teste objetivo). Os mtodos de pesquisa para clculo das jogadas a realizar que permitam ao computador jogar sozinho e resolver os puzzles devem ser descritos na seco seguinte assim como o mtodo geral para os chamar e resolver o puzzle (utilizando um dado mtodo selecionado de entre os disponveis).

VI. Search Algorithms

Descrevendo os vrios algoritmos de pesquisa utilizados e a sua implementao de modo a calcular a prxima jogada do PC ou retornar a soluo final (conjunto de operaes para transformar o estado inicial no estado objetivo). Devem ser implementados algoritmos para clculo da soluo utilizando pesquisa em largura, pesquisa em profundidade (se aplicvel), aprofundamento progressivo, custo uniforme (se aplicvel), pesquisa gulosa e Algoritmo A* (estes ltimo mtodo utilizando vrias heursticas).

VII. Experiments (Tests?) and Results

Descrevendo as experiłncias realizadas com os vrios algoritmos para resolver diversos puzzles e os resultados obtidos a nvel de tempo e custo da soluo obtida em cada nvel, por cada um dos mtodos experimentados. Devem ser includas tabelas comparativas dos resultados obtidos na aplicao dos vrios mtodos aos vrios puzzles (nveis do jogo) e discutidos os resultados.

VIII. Conclusions and Future Work

As of yet, the problem has been completely formalized - it has been formally described, and formulated as a search problem (a Game State representation has been reached, the initial and goal states have been described and the operators and path cost have been stated).

The chosen programming language was Python, and a simple graphical interface has been implemented, in order to show the solutions of the different algorithms when they are implemented.

In order to study the best possible approach for solving *Unblock Me* puzzles, the following algorithms will be the subject of various tests:

- Breadth-First Search
- Depth-First Search
- Iterative Deepening
- Uniform Cost Search
- Greedy Search
- A*
- Bi-Directional Search (using Greedy Search)
- Bi-Directional Search (using A*)

For the informed search algorithms (Greedy and A*), several different heuristics will be experimented with, such as:

- The number of pieces between the special piece and the level exit
- The number of cells (distance) between the special piece and the level exit

In order to evaluate the implemented algorithms, they will be assessed both in terms of the achieved solution's quality and in terms of the algorithm's performance. For the first metric, the main evaluation point will be the number of piece movements. For the latter, both the execution time and the number of evaluated states will be taken into account.

Sumrio do trabalho realizado e concluses que retira deste projeto. Anlise crtica dos resultados obtidos em comparao com os resultados tericos que seriam esperados. Trabalho futuro, ou seja, formas de melhorar o trabalho desenvolvido.

References

[1] "Unblock me FREE.". Google Play. February 28, 2019.https://play.google.com/store/apps/details?id=com.kiragames.unblockmefree.

[2] Fogleman, Michael. ”Solving Rush Hour, the Puzzle.”. July, 2018. https://www.michaelfogleman.com/rush/.
[3] ”Bitboard.”. Wikipedia - The free Encyclopedia. December 6, 2018. https://en.wikipedia.org/wiki/Bitboard.
[4] Littman, Michael. ”Programming Assignment P1 - What A* Rush”. Priceton. 2012. https://www.cs.princeton.edu/courses/archive/fall12/cos402/assignments/programs/rushhour/.
[5] Findling, Rainhard. ”The RushHour Puzzle an Artificial Intelligence Toy Problem.”. April 4, 2012. http://geekoverdose2.rssing.com/browser.php?indx=39804402&item=1.