Search algorithms comparison

(Roll the Block (**Bloxorz**) /Número do Grupo)

Filipa Durão (up201606640)
FEUP, MIEIC
Porto, Portugal
up201606640@fe.up.pt

Pedro Fernandes (up201603846)
FEUP, MIEIC
Porto, Portugal
up201603846@fe.up.pt

Pedro Silva (up201604470)
FEUP, MIEIC
Porto, Portugal
up201604470@fe.up.pt

***Abstract*— In this report the group will describe the development ongoing to the implementation of the game Bloxorz. The game consists on a block that must roll on a map to reach a goal. The language chosen for the implementation is Python. The main goal of the present work is to implement several search algorithms to solve the problem and compare their efficiency.**

Keywords—Artificial Intelligence, Search Algorithms

I. Introduction

This work was developed to the course of IART (Artificial Intelligence). On this report there will be a brief and non-technical description of the puzzle to be solved on chapter **II - Description of the problem**. Afterwards, on chapter **III - Formulation of the problem** there is a technical description of the puzzle, but that doesn't involve any details about the implementation. On chapter **IV - Related Works** there is a brief description on what the group found about the puzzle's previous implementations by other parties. In the last chapter, chapter **V - Conclusions and development perspectives**, there is a brief description of the conclusions the group got from the present project and possible future work to be made to improve it.

II. Description of the problem

The problem "Roll the Block" consists in a block of size 1x1x2 that rolls over itself in any direction within a 2d plan.

In that plan there is a map, the block must roll without going outside its borders. The goal is to take the block to a finish cell, where it must stand vertically in order to finish the level. The goal is to solve the level with the minimum number of moves.

III. Formulation of the problem

The map is represented by a matrix of characters representing the block, free spaces inside the map, the finish square and the space outside the map. The block always has 2 coordinates, because it is 2 units long. However, this 2 coordinates might be the same, meaning the block is standing up. A state is represented by the coordinates of the block in the map.

The map configuration is loaded from a file and the finish cell position and the initial position of the block are deduced from it. The map representation is stored in a constant matrix all throughout the process. The initial state is the initial position of the block.

The final state of the process is when both of the coordinates of the block match the coordinate of the finish cell, meaning that the block is standing up on top of the finish.

There are 12 possible operators:

- Up when the block is standing
  - **pre-condition:** block is standing
- Up when the block is laying on the y axis
  - **pre-condition:** block is laying on the y axis
- Up when the block is laying on the x axis
  - **pre-condition:** block is laying on the x axis
- Left when the block is standing
  - **pre-condition:** block is standing
- Left when the block is laying on the y axis
  - **pre-condition:** block is laying on the y axis
- Left when the block is laying on the x axis
  - **pre-condition:** block is laying on the x axis
- Right when the block is standing
  - **pre-condition:** block is standing
- Right when the block is laying on the y axis
  - **pre-condition:** block is laying on the y axis
- Right when the block is laying on the x axis
  - **pre-condition:** block is laying on the x axis
- Down when the block is standing
  - **pre-condition:** block is standing
- Down when the block is laying on the y axis

    - **pre-condition:** block is laying on the y axis
- Down when the block is laying on the x axis
    - **pre-condition:** block is laying on the x axis

The cost of every operator (roll) is 1.

The final cost of the solution is the number of moves performed, meaning, is the sum of all move costs (which are 1). The goal is to solve the problem in the least number of moves.

IV. Related works

During the research stage of the project, the group came across several implementations of algorithms that are that can be adapted and used on our own project, such as Python implementations of the BFS, DFS, A* and Dijkstra algorithms. The group also found several implementations of the game in multiple languages. In this report, we will mention only the most relevant ones for our implementation:

- **bloxorz_solver** by alppboz : a github repository with an implementation of the bloxorz game that has a solver included, using the A* algorithm. [1]
- **aima-python** by aimacode : a github repository with Python implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach". [2]
- **Implementation of A*** by Red Blob Games : a website featuring several algorithm implementations in different languages, including an A* implementation in Python. [3]

V. Conclusions and development perspectives

On this first stage of the project all goals were fulfilled. The group has made extensive research on the topic assigned, both the puzzle and it's operations and the algorithms that will be implemented for case studies, and how to implemented in the language chosen, Python. In the future developments of this project, the group aims to implement the game with several distinct search algorithms to solve it, and then make a analysis of the efficiency of each one of the said algorithms.

References

[1] alppboz, "bloxorz_solver",
URL: https://github.com/alppboz/bloxorz_solver, visited on March 17th
[2] aimacode, "aima-python",
URL: https://github.com/aimacode/aima-python, visited on March 17th
[3] Red Blob Games, "Implementation of A*", URL: https://www.redblobgames.com/pathfinding/a-star/implementation.html, visited on March 14th