
Cogito
Heuristic Search Methods for Problem-Solving
COGITO
Press Left Button To Continue
Press Right Button For Menu.
Bernardo Campos - up202006056
Davide Teixeira - up202109860
Emanuel Maia - up202107486

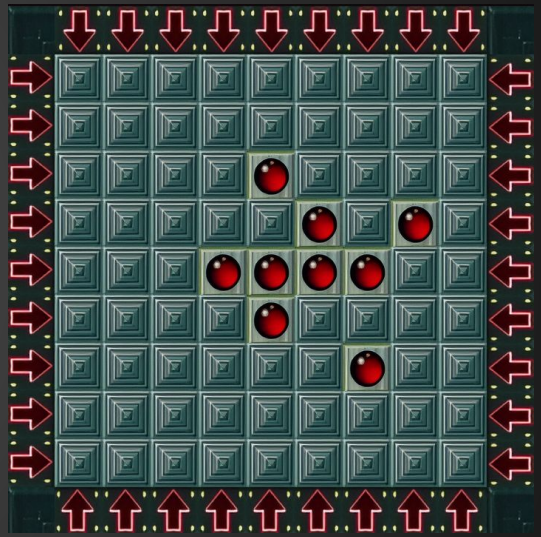
Game Definition

Cogito is a single player puzzle game in a randomly shuffled board.

The objective is to rearrange the pieces on the board to replicate the positions specified on a provided board.

Each level has one of 12 rules, that change how the actions of the player move the board.

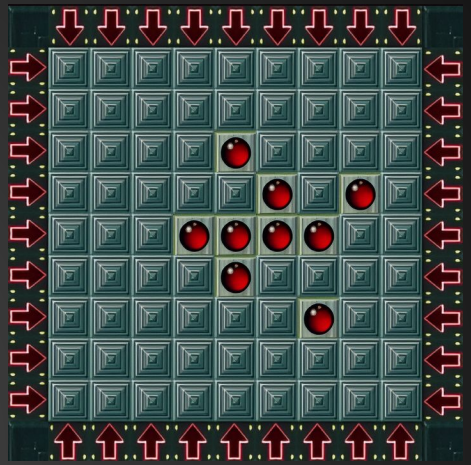
Completing the puzzle allows the player to move to the next level, with a different board and rule.



Problem Representation

State Representation

- The state of the game can be represented as the board of the game itself.
- In a computational way, that board can be represented as a matrix of boolean cells, in which a cell can, or not, have a ball.



Initial State

- The initial state of the game is random. In each level, several random moves are applied to the board in order to reach a random initial state.

Objective State

- The objective state is the state in which the playing board matches the given objective board.

Operators

The operators can be defined as the click on one of the arrow tiles that can shift the rows or columns of the board a number of times.

The operator can be represented as a tuple (x, y) that associates for each button, the information for shifting/choosing a specific row/column.

There are no preconditions for this operator to work.
The exact effect of the operator will vary depending on the current level, but it always results on shifting a row or column a number of times.

The goal is to complete the level with the least amount of moves possible. With than in mind, the score can be constant for each operator (cost = 1).

Implemented Search Algorithms

Uninformed Search

- Iterative Deepening - performs DFS with limited depth, iteratively increasing this depth.
- Uniform Cost - expands the frontier node with the lowest cost,

Heuristic Search Methods

- Greedy Search - expands the node that, according to a defined heuristic, seems to be closer to a solution.
- Weighted A* Algorithm - similar to the Greedy Search algorithm, but takes into consideration the cost to reach the a state. Use of weighted alternative sacrifices possibly better solutions for space/time.

Heuristic Functions

Three heuristic evaluation functions were defined:

- Heuristic 1
  - The board is evaluated based on the number of pieces that are inside of the correct position.
- Heuristic 2
  - Sum of the distances between each piece and the closest correct position.
- Heuristic 3
  - Sum of the distances of each piece to the center of the board.

Experimental Results - Uninformed Methods

We could not get viable results with our implemented Uninformed Search Algorithms.

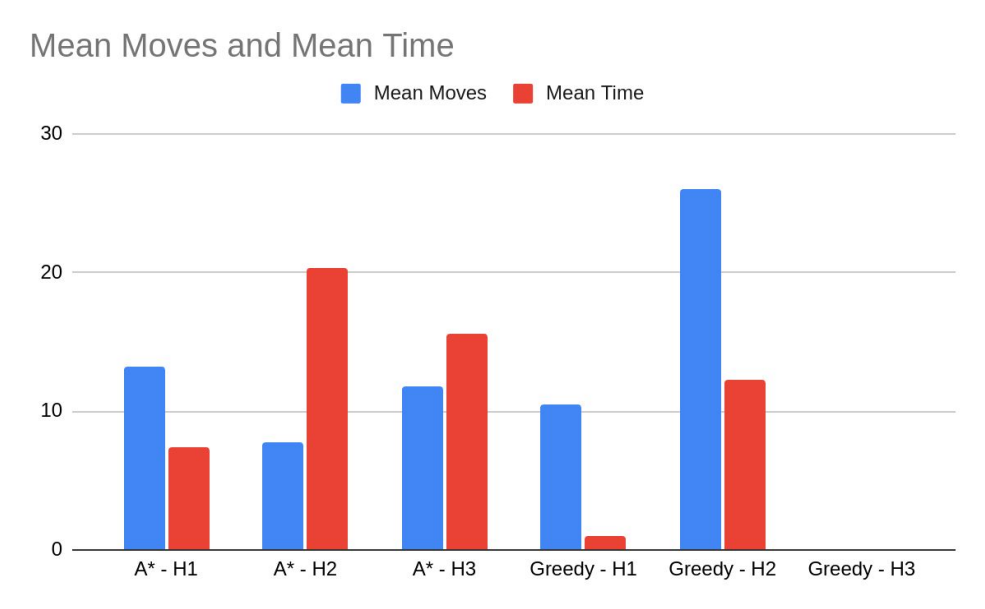
These algorithms have high time and space complexities, combined with the game’s high branching factor (minimum of 12 and maximum of 36) finding a solution in viable time is not possible.

Experimental Results - Informed Methods

We tested the efficiency of our informed algorithms by tracking the moves and time necessary to complete each of the first 12 levels. For standardization purposes, for each level, the same randomized board was used when measuring an algorithm and heuristic.

The Greedy Algorithm, for most levels was only able to find a solution using H2. We considered an alternative not viable when it took >60s to compute the first move.

When comparing the A* alternatives, we found that H2 was consistently the heuristic using less moves, and H1 the heuristic executing in faster time.



Conclusions

In this assignment, we learned how to put into practice the knowledge given to us during the theoretical and practical classes.

We made a formal definition of the game as a search problem, including its state representation, initial state, objective tests and operators.

We implemented and compared the efficiency of different search algorithms, as well as developed various heuristic evaluation functions, and in the end we were able to achieve a viable way of having the computer solving the game.

References

- Available course resources
- Cogito (1992, Macintosh) - Levels 1~18 (Youtube Gameplay by Rubycored)
- https://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

Materials

Software and Languages:

- Python and Pygame

Slides and Graphics:

- Google Slides and Google Sheets