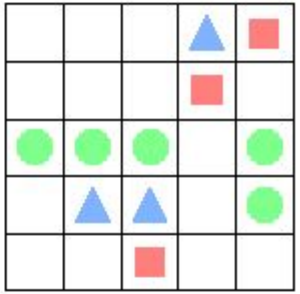
Symmetry Puzzles

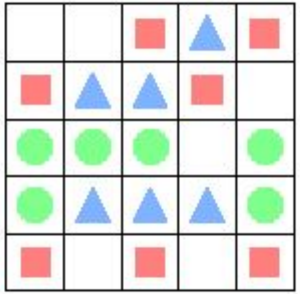
IA (2022/2023)

Projeto Realizado Por:
João Sousa
Mónica Araújo
Mónica Moura Pereira

Specification

- Game Definition/Objective:
    - You can put shapes into an empty square;
    - Ignoring the blank spaces, the shapes in each row and column should be palindromes;
    - Each puzzle has an unique solution.





Related Work

- Field of research the game is in: Computational Symmetry
  - https://www.cse.psu.edu/~yul11/0600000008-Liu.pdf
- Other similar games : Sudoku puzzles, Symmetry Shuffle
- A* algorithm
  - http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
  - https://www.geeksforgeeks.org/a-search-algorithm/

Formulation of the Problem

- **State Representation**: Represented by a Matrix (0: no piece; 1+: shapes)
- **Initial State:** Partially filled board with some shapes already correctly place and the others empty.
- **Objective Test:** All columns and all rows are palindromes, ignoring the blank spaces.
- **Operators**:
  - putPiece (position, piece_type)
    - <u>Preconditions</u>:
      - board[position] == 0;
      - position is valid;
      - piece_type is already present in either the row or column.
    - <u>Effects</u>: matrix(position) = piece_type;
    - <u>Costs</u>: Each step costs 1, the cost of the solution if the total amount of steps;

```
 Initial State:                Objective State:

[[0, 0, 0, 1, 2],          [[0, 0, 2, 1, 2],
 [0, 0, 0, 2, 0],           [2, 1, 1, 2, 0],
 [3, 3, 3, 0, 3],    =>     [3, 3, 3, 0, 3],
 [0, 1, 1, 0, 3],           [3, 1, 1, 1, 3],
 [0, 0, 2, 0, 0]]           [2, 0, 2, 0, 2]]
```

```
 Initial State:             PutPiece( (1,1), 1):

[[0, 0, 0, 1, 2],          [[0, 0, 0, 1, 2],
 [0, 0, 0, 2, 0],           [0, 1, 0, 2, 0],
 [3, 3, 3, 0, 3],    =>     [3, 3, 3, 0, 3],
 [0, 1, 1, 0, 3],           [0, 1, 1, 0, 3],
 [0, 0, 2, 0, 0]]           [0, 0, 2, 0, 0]]
```

Formulation of the Problem

- **Heuristics/Evaluation:**
    - h1(n) - number of rows and columns that aren’t palindromes / 2;
    - h2(n) - determines how close each row and column are to being palindrome;
    - h3(n) - makes an estimation of how many pieces have to be put to end the game;
    - g(n) - total cost of reaching current state n from initial state.

Implemented Algorithms

- Uninformed Search Methods
  - BFS
  - DFS
  - Iterative-Deepening
  - Uniform Cost (the same as BFS because our cost to make a move will be 1)
- Heuristic Search Methods
  - Greedy
  - A*
  - Weighted A*

Results - Comparing all algorithms

- Divided Games into 3 difficulties:
    - Easy Games
        - All algorithms worked relatively well except of DFS
    - Normal Games
        - Uninformed Search Methods don't work well as search tree is significantly bigger
        - Heuristic Methods continue to have good performance
    - Hard Games
        - Greedy and A* start to fail, but Weighted A* maintains good performance

EASY GAMES

Algorithms	time	steps
BFS	1,196367264	630
Iterative Deepening	0,06929540634	339
Greedy	0,03143191338	12
A*	0,04267597198	9
Weighted A*	0,02911400795	6

NORMAL GAMES

Algorithms	time	steps
Greedy	0,241538763	9
A*	2,707150698	60
Weighted A*	0,234431982	9

HARD GAMES

Algorithms	time	steps
Weighted A*	0,5554494858	15

Results - Heuristics Performance on Normal Difficulty games

Fig. 1 - Time Performance of Heuristic Functions on normal difficulty games (greedy and weighted A*)

NORMAL GAMES

Greedy	time	steps
h1	0,461676836	30
h2	0,2596170902	16
h3	0,5895299911	25

NORMAL GAMES

Greedy	time	steps
h1	0,64569664	35
h2	0,3949882984	20
h3	0,4122750759	17

Results - Heuristics Performance on Hard Games

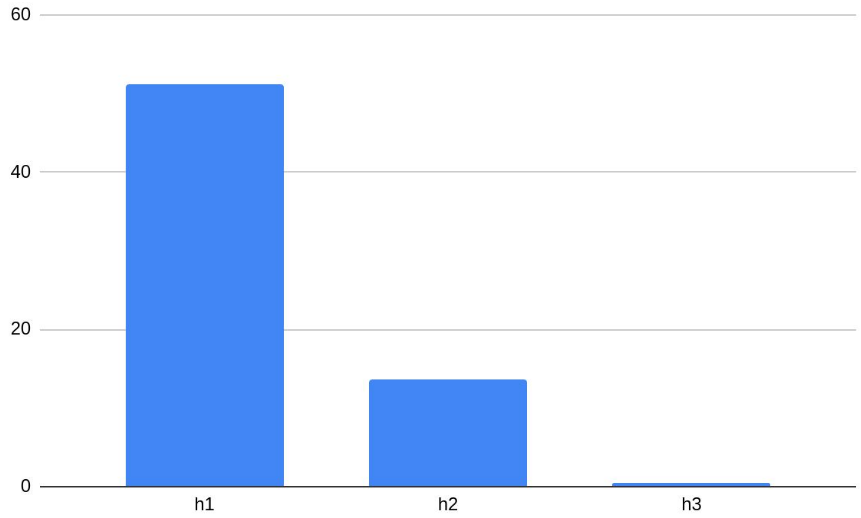


Fig. 2 - Time Performance of Heuristic Functions on hard games (weighted A*)

HARD GAMES	Weighted A*	
Heuristics	Time	Steps
h1	51,22115588	864
h2	13,66801548	268
h3	0,4655964375	15

Conclusions

- Best Performing Algorithm
  - Weighted A*
- Best Performing Heuristics
  - On normal games best heuristic is h2
  - On hard games best heuristic is h3
- Worst Performing Algorithms
  - BFS
  - DFS (goes in a loop and doesn't find any solutions)