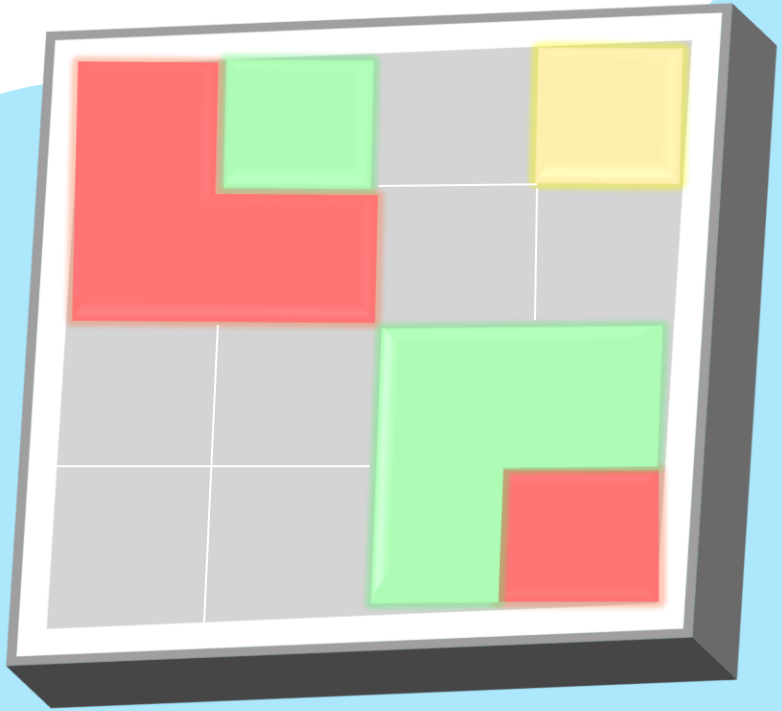
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Cohesion

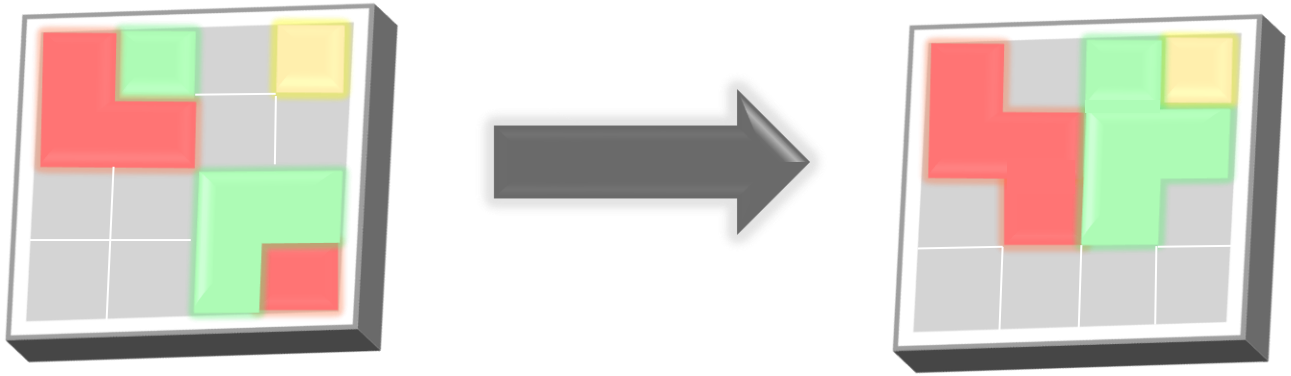


Artificial Intelligence
2022/2023 3LEIC10 Group 27

Daniela Tomás up202004946
Diogo Nunes up202007895
João Veloso up202005801

Problem Specification

Cohesion is a puzzle game where the player slides squares of different colors, and when two squares of the same color are placed next to each other, they bond together and become permanently attached. The puzzle is solved when all the squares of each color are connected.



Formulation of the problem

- **State Representation:** Matrix MxN composed of squares with different colors
- **Initial State:** Generated board and squares
- **Objective Test:** Check wether all the squares are connected to all other squares of the same color
- **Operators:**

Names	Preconditions	Effects	Costs
moveUp	There is an empty space above the square and all attached pieces of the same color will remain connected after the move	The square and all attached pieces move up by one position	1
moveDown	There is an empty space below the square and all attached pieces of the same color will remain connected after the move	The square and all attached pieces move down by one position	1
moveLeft	There is an empty space left to the square and all attached pieces of the same color will remain connected after the move	The square and all attached pieces move left by one position	1
moveRight	There is an empty space right to the square and all attached pieces of the same color will remain connected after the move	The square and all attached pieces move right by one position	1

Formulation of the problem (cont.)

- **Heuristics:**
  - **Heuristic 1 – Distance:**
    - **Heuristic 1.1 –** Calculate the distance between all elements of the same color
    - **Heuristic 1.2 –** Calculate the Manhattan distance between each square and its closest neighbor of the same color
  - **Heuristic 2 – Number of squares:**
    - **Heuristic 2.1 –** Calculate the number of groups of squares for each color that are not yet connected to each other
    - **Heuristic 2.2 –** Calculate the number of squares that are not yet connected and adds up their Manhattan distance

Work already done

- **Programming language:** Python, using Pygame for the graphic interface
- **Development environment:** VSCode
- **Data structures:** Custom classes for different entities + more to come

- The game development is already in progress

- Demo link:

  https://drive.google.com/file/d/1vmbYidO_sJHE37yMsZ9yVJeQC7nVtCPJ/view?usp=sharing

Algorithms

- We implemented the following algorithms:
  - **Uninformed:**
    - Breadth-First Search
    - Depth-First Search
    - Uniform cost
    - Iterative deepening
  - **Informed:**
    - Greedy
    - A*
- Approach:
  - For heuristics, we choose to implement 2 different ones, namely:
    - **Manhattan distance:** For this heuristic, we sum the Manhattan distances of all blocks of the same color.
    - **Color clusters:** For this heuristic we calculate the minimum distance between any 2 pieces of differing colors.

Results

		Algorithm							
		BFS	DFS	Uniform Cost	Iterative Deepening	A* Manhattan	Greedy Manhattan	A* Color Cluster	Greedy Color Cluster
Level 1 (4x4)	Time (s)	0,546	0,568	1,043	0,3	0,018	0,017	1,213	0,182
	Moves	5	5	5	5	5	5	5	16
	Nodes	310	321	715	622	6	6	872	104
Level 2 (4x4)	Time (s)	0,897	0,772	1,139	0,629	0,091	0,048	2,312	-
	Moves	5	5	5	5	8	8	5	-
	Nodes	187	181	278	443	18	9	702	-
Level 3 (6x6)	Time (s)	108,15	60,117	-	426,257	0,058	0,051	-	-
	Moves	10	10	-	10	10	10	-	-
	Nodes	60969	48250	-	1048269	16	12	-	-
Level 4 (6x6)	Time (s)	-	-	-	-	0,625	-	-	-
	Moves	-	-	-	-	26	-	-	-
	Nodes	-	-	-	-	63	-	-	-

Results

- Even in **level 1**, the simplest of all, we can already see some differences between uninformed and informed search algorithms, as when they are guided by a good heuristic for this instance, in this case, the Manhattan distance, as there are only blocks of 1 color with no obstacles only 6(!) nodes are explored to find the optimal solution, whereas uninformed searches take between 300 to 700 (*Figure 2*).
- In **level 2**, which has more pieces and of different colors, we can see that informed search algorithms guided by Manhattan distance can´t find the same optimal solution that uninformed ones do, but they do find this solution in a much lower number of nodes explored, which will have more clear advantages on the next levels.
- By **level 3**, which is on a significantly bigger board, uninformed searches are starting to take quite a bit more time, the most noticeable of which is iterative deepening which goes from an average of 0,629 seconds to just over 7 minutes (exploring over 1 000 000 nodes). Meanwhile A* and greedy both find a solution extremely quickly, exploring less than 20 nodes each.
- By **level 4**, only A* guided by Manhattan distance manages to find a solution, and although it is fast finding it, it is not an optimal one, which is estimated at around 16 moves.

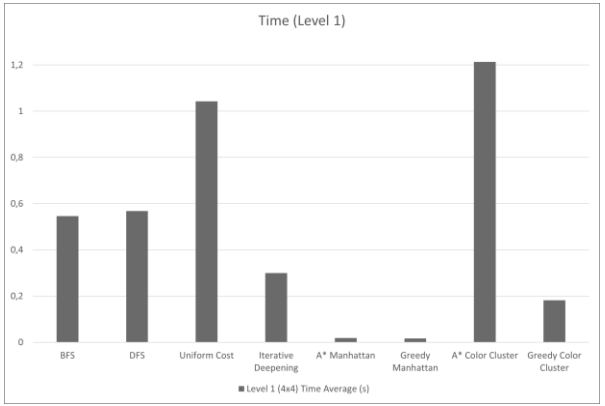


Figure 1: Time (Level 1)

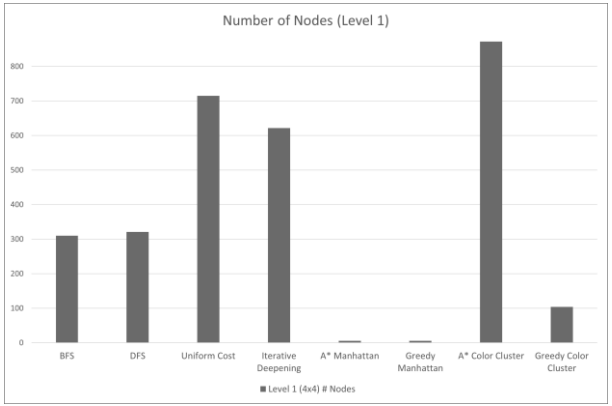


Figure 2: Number of Nodes (Level 1)

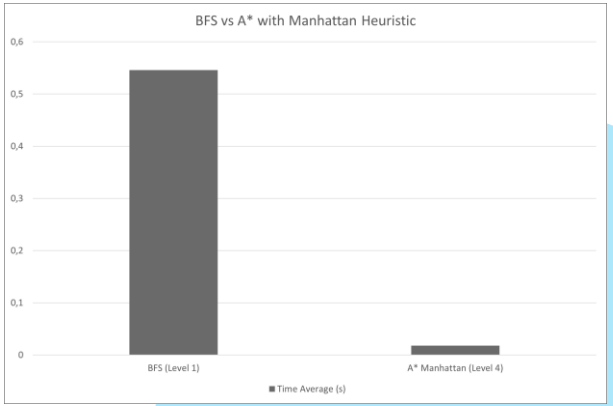


Figure 3: Time - BFS (Level1) vs A* Manhattan (Level 4)

Conclusions

- Informed searches are superior in bigger boards as they find a solution in useful times, whereas uninformed ones would simply take too long. They also show potential to being superior in smaller boards, if guided by an admissible (never overestimates the actual cost) and consistent heuristic function, which we didn't manage to find for this game.

References

- Google Play - Cohesion Free
  https://play.google.com/store/apps/details?id=com.NeatWits.CohesionFree&hl=en&gl=US
- Solver implementation for the 15 puzzle
  https://github.com/MichaelKim/15puzzle
- Evaluating Search Algorithms for Solving n-Puzzle
  http://sumitg.com/assets/n-puzzle.pdf
- Stack Overflow
  https://stackoverflow.com/questions
- Pygame Documentation
  https://www.pygame.org/docs/