Project 1 Report

Rajul Bothra & Dennis Yu

Table of Contents:

Task 1: Problem Formulation

Environment	N by N grid
State Space	N by N Array of Nodes
Actions	a valid move on grid
Perception/Observation	current position on the grid
Transition Function	move piece how many cells in a direction according to number in cell
	update state by moving current cell to destination cell
Evaluation Metric	Positive evaluation: if goal is reached
	Negative evaluation: if goal can't ever be reached

Task 2: Puzzle Generation and Representation

Show input received, puzzle generated for n = 5, 7, 9, 11

```
Enter a number for n:
5
Gameboard
3 2 1 2 1
3 2 2 3 4
1 1 2 2 1
2 3 3 1 2
1 2 3 2 0
```

```
Enter a number for n:
7
Gameboard
1 1 2 2 1 5 6
5 3 1 4 5 3 6
6 3 1 1 2 1 3
1 2 3 2 4 5 2
2 4 4 4 2 5 6
3 3 4 4 2 5 5
1 1 6 6 2 1 0
```

```
Enter a number for n:
9
Gameboard
3 8 8 4 2 6 6 1 7
8 1 2 2 6 2 2 4 4
1 3 1 6 1 4 2 1 4
8 6 2 4 4 4 5 5 2
6 4 3 3 3 1 6 2 3
2 5 3 3 4 4 5 2 2
7 7 3 1 3 4 4 4 2
4 1 6 2 2 7 3 5 1
3 1 2 2 3 6 6 5 0
```

```
Enter a number for n:
11
Gameboard
5 3 6 7 8 3 1 2 8 9 10
10 7 3 8 3 8 2 3 9 4 8
10 3 8 2 6 1 5 1 6 9 4
2 5 6 3 1 2 5 2 5 5 4
2 6 7 2 3 5 4 6 3 5 6
9 7 5 6 3 5 6 1 2 5 7
10 5 6 5 5 4 1 1 3 9 9
4 5 7 7 1 2 7 7 2 9 1
5 5 2 5 8 1 6 8 2 2 4
2 3 1 7 5 9 6 1 6 8 9
9 8 1 8 3 3 2 8 2 9 0
```

Task 3: Puzzle Evaluation

2 example puzzles each for n = 5, 7, 9, 11

```
Enter a number for n:
5
Gameboard
3 2 1 2 1
3 2 2 3 4
1 1 2 2 1
2 3 3 1 2
1 2 3 2 0
Value: -3
Evaluation grid
0 2 3 1 X
2 4 4 3 5
4 3 4 2 5
1 4 2 X 5
3 4 5 3 X
```

```
Enter a number for n:
5
Gameboard
4 3 1 3 4
3 1 1 3 3
2 3 1 2 4
4 1 2 2 2
4 2 4 3 0
Value: 2
Evaluation grid
0 X X X 1
X X X X X
X X X X X
X X X X X
1 X X X 2
```

```
Enter a number for n:
7
Gameboard
1 6 2 2 3 5 3
2 3 5 1 5 2 3
2 3 3 2 2 1 1
3 2 1 1 3 4 4
2 5 2 4 1 3 1
2 3 3 3 2 2 6
1 4 2 1 4 6 0
Value: -4
Evaluation grid
0 1 X 5 5 4 X
1 6 2 7 7 6 6
6 3 5 4 4 5 5
2 5 4 3 4 6 6
5 7 4 4 5 6 7
4 4 5 6 5 5 6
3 2 3 X 4 3 X
```

```
Enter a number for n:
7
Gameboard
1 1 2 2 1 5 6
5 3 1 4 5 3 6
6 3 1 1 2 1 3
1 2 3 2 4 5 2
2 4 4 4 2 5 6
3 3 4 4 2 5 5
1 1 6 6 2 1 0
Value: 5
Evaluation grid
0 1 2 4 3 4 7
1 2 3 4 3 2 X
4 4 3 4 5 X 5
6 6 4 5 6 5 X
4 3 5 X 5 3 6
3 4 6 4 5 5 6
2 3 4 X 4 X 5
```

```
Enter a number for n:
9
Gameboard
3 8 8 4 2 6 6 1 7
8 1 2 2 6 2 2 4 4
1 3 1 6 1 4 2 1 4
8 6 2 4 4 4 5 5 2
6 4 3 3 3 1 6 2 3
2 5 3 3 4 4 5 2 2
7 7 3 1 3 4 4 4 2
4 1 6 2 2 7 3 5 1
3 1 2 2 3 6 6 5 0
Value: 5
Evaluation grid
0 4 X 1 X 5 3 2 3
5 4 5 3 4 4 X 3 3
X 5 7 5 6 7 5 6 6
1 4 6 4 7 5 3 5 2
3 9 X 2 8 7 3 9 6
5 5 6 4 6 5 4 4 3
X 5 5 6 7 6 4 8 5
5 4 5 3 5 4 6 5 4
5 5 6 5 7 6 4 6 5
```

```
Enter a number for n:
9
Gameboard
6 2 7 1 1 8 7 2 1
8 1 5 4 7 4 1 4 5
6 5 2 6 5 2 1 1 4
3 4 2 3 2 1 6 6 8
6 7 4 5 1 1 3 3 6
6 3 4 1 3 5 6 1 8
3 3 5 5 4 6 2 3 1
6 2 2 3 2 7 3 2 5
2 3 7 1 5 8 4 8 0
Value: -14
Evaluation grid
0 7 5 4 5 6 1 X X
5 6 6 3 6 5 4 4 6
X 7 X 7 6 6 5 6 5
2 6 8 3 7 7 4 5 X
X X 6 4 7 7 3 6 5
5 6 5 4 5 6 6 5 4
1 X 7 2 5 X X 4 3
4 7 6 3 7 8 2 7 4
X 7 7 8 6 7 X X X
```

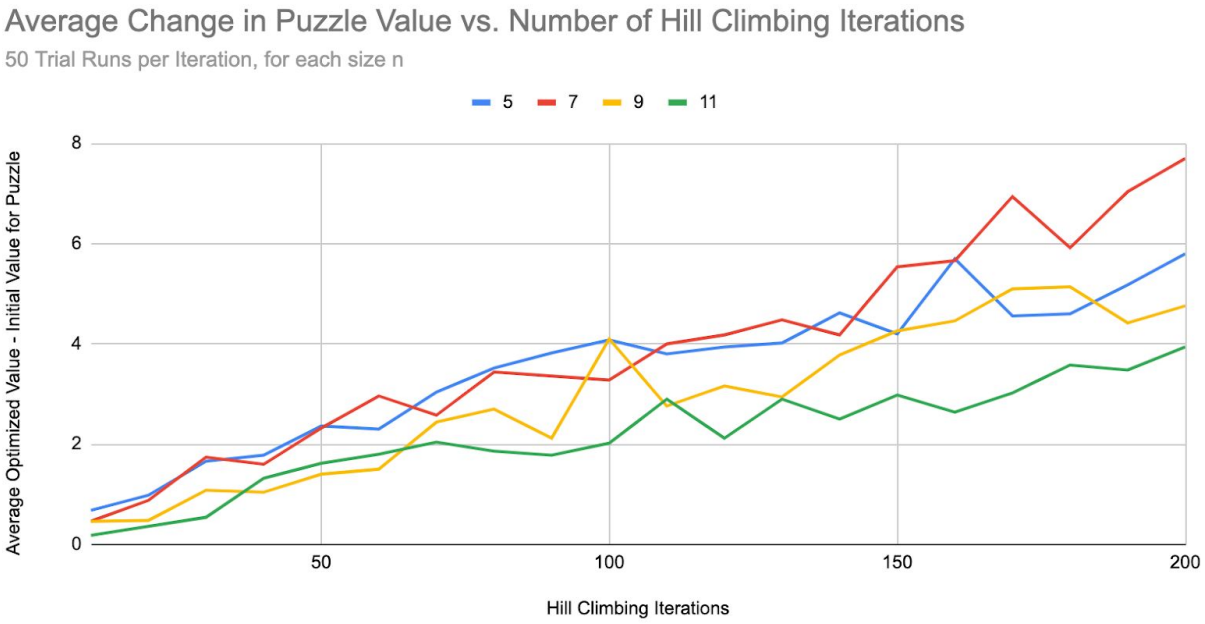
```
Enter a number for n:
11
Gameboard
3 7 3 2 9 5 2 6 2 2 9
3 7 4 2 9 9 7 3 7 8 1
3 2 2 2 2 7 7 5 4 7 4
1 8 1 3 1 7 2 2 7 1 3
4 9 4 2 1 2 4 2 7 5 1
4 8 1 4 4 3 5 2 3 8 5
8 9 2 2 4 3 6 6 4 1 1
6 8 7 3 5 1 1 1 7 6 10
7 3 2 4 6 5 5 5 5 3 1
7 9 3 1 2 9 1 4 9 6 10
10 1 8 4 5 9 7 9 10 2 0
Value: 7
Evaluation grid
0 2 6 1 5 2 4 7 3 X 3
4 6 X 5 5 6 7 6 7 5 7
2 3 5 2 5 3 5 6 4 4 8
1 2 7 5 4 5 6 5 4 3 4
2 4 5 3 3 4 6 5 X 4 5
3 5 4 5 4 3 7 6 4 6 6
6 5 5 4 6 5 7 6 5 6 5
6 3 X 5 6 7 6 7 8 4 6
3 7 5 5 6 4 7 4 5 6 5
4 X 6 6 5 4 6 5 X 5 4
X X 6 6 6 6 7 7 5 7 7
```

```
Enter a number for n:
11
Gameboard
7 3 2 7 3 7 4 7 10 4 1
5 4 9 7 2 9 9 8 4 8 4
4 8 3 2 7 6 3 7 6 7 5
6 5 4 2 2 3 7 3 1 2 1
9 8 2 3 5 3 3 7 6 6 7
2 7 6 3 2 4 6 5 6 4 10
9 3 5 1 4 3 4 3 4 5 8
9 7 8 1 5 3 3 3 7 9 4
8 2 3 4 7 1 3 3 8 7 9
3 1 1 6 9 5 7 7 5 9 10
4 6 3 8 8 1 10 9 5 6 0
Value: -8
Evaluation grid
0 7 7 9 6 9 6 1 10 8 7
X 7 8 7 8 8 6 4 X 6 8
7 9 8 8 4 6 X 6 9 10 5
9 8 8 8 7 9 8 6 10 5 4
4 11 10 6 11 9 5 3 10 5 5
8 8 9 7 8 7 8 8 9 6 9
7 10 7 6 7 10 X 7 8 8 8
1 10 6 5 3 8 4 2 9 2 3
11 9 10 6 8 7 8 7 10 9 8
6 11 12 7 5 8 X 5 X 7 9
X 10 9 7 8 9 5 3 9 6 X
```

Task 4: Hill Climbing

Provide a plot of how the evaluation function changes as the number of iterations increases averaged over multiple runs of the approach (at least 50). Report these statistics for different sizes of puzzles (n = 5,7,9,11).

The data used to generate the graph is attached in the spreadsheet.



Task 5: Shortest Path First

We implemented BFS for SPF, so the code is very similar to the BFS we used for the evaluation function from Task 3.

Task 6: A*

- Define an A* heuristic for this problem and implement the A* algorithm with this heuristic.
    - If the cell is associated with a path that leads to the goal, it is assigned a heuristic.
    - That heuristic is how many counts/cells are needed to be taken to reach the goal.
    - If the cell does not have a path that connects to the goal, it's heuristic is -1.
- How does the computation time compare to that of SPF and what problems can A* solve that SPF cannot?

- The implementation of A* does reduce the amount of actions to reach the goal. However, the runtime is way longer than SPF. We think this is due to inefficiencies caused by our implementation of a max priority queue.

Task 5 & 6 Analysis:

```
Enter a number for n:
5
Gameboard
4 4 3 2 1
3 1 2 2 4
3 3 2 3 4
2 2 1 3 2
2 2 2 4 0
Value: 3
Evaluation grid
0 3 4 2 1
3 5 6 4 2
2 5 3 3 4
6 6 5 5 X
1 4 2 5 3
SPF: 26299 nanoseconds
SPF
0 3 X 2 1
3 X X X 2
2 X 3 3 X
X X X X X
1 X 2 X 3
A*: 1049500 nanoseconds
a*
0 X X X 1
X X X X X
2 X 3 X X
X X X X X
1 X 2 X 3
```

```
Enter a number for n:
5
Gameboard
1 1 3 2 4
3 1 3 3 1
3 3 2 2 3
1 3 1 3 4
3 4 2 3 0
Value: 5
Evaluation grid
0 1 2 5 X
1 2 3 2 X
5 3 4 6 4
5 4 3 4 5
2 X 4 3 5
SPF: 26401 nanoseconds
SPF
0 1 2 5 X
1 2 3 2 X
5 3 4 X 4
5 4 3 4 5
2 X 4 3 5
A*: 1112600 nanoseconds
a*
0 1 2 X X
1 2 3 X X
X 3 5 X X
X X X X X
5 X 4 X 5
```

```
Enter a number for n:
5
Gameboard
2 1 3 1 2
2 2 3 1 2
1 3 2 3 2
3 2 2 3 3
2 4 3 3 0
Value: 4
Evaluation grid
0 X 1 4 4
2 5 3 5 6
1 2 4 6 3
2 4 2 3 3
X X 4 X 4
SPF: 37600 nanoseconds
SPF
0 X 1 4 4
2 X 3 5 X
1 2 4 X 3
2 4 2 3 3
X X 4 X 4
A*: 963799 nanoseconds
a*
0 X 1 X 4
2 X X X X
1 2 4 X 3
2 X X X X
X X X X 4
```

Task 7: Population Algorithm

You are requested to describe in detail the population-based local search approach that you have developed in order to search this puzzle generation challenge. Any parameters that you need to define in order to execute the algorithm (e.g.,size of population, probability of selection) need to be explicitly mentioned and evaluated in terms of their impacts. Justify the choices you made in terms of the algorithm you decided to implement.

Initial population

Start with 4 random boards: {A', B', C', D'} - we chose this because it is small enough that algorithm is not confusing, but large enough to allow for some variation in boards.

Fitness Function

Evaluate from Task 3

Evolve - repeat these steps for number of iterations

1. Selection of 2 best from population of 4 boards: {A', B', C', D'}
    - A - best evaluation
    - B - second best evaluation
2. Generate 2 with crossover of 2 best
    - C = front half of A, back half of B - create a crossover with the two best boards
    - D = front half of B, back half of A - create a different crossover with the two best boards
3. Select the 2 fittest from population of 4 boards: {A, B, C, D}
    - A' - best evaluation - keeps the original boards or the better crossover
    - B' - second best evaluation - keeps the original boards or the better crossover
4. Add 2 random ones
    - C' - random - to add some variation to the population
    - D' - random - to add some variation to the population

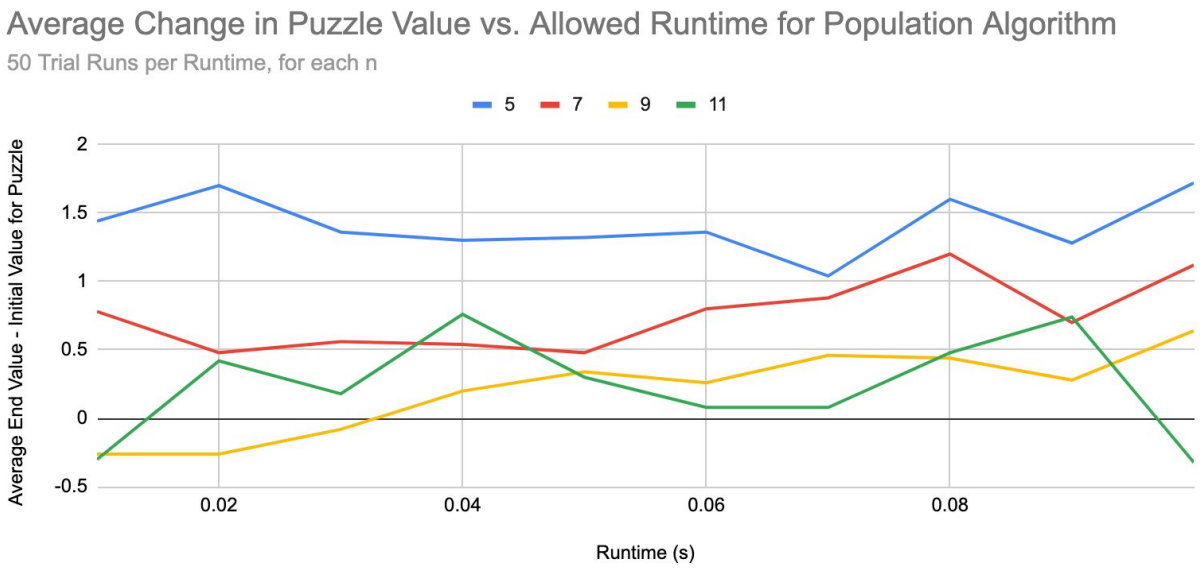
Function Output

Final puzzle
Value
Iterations
Computation time

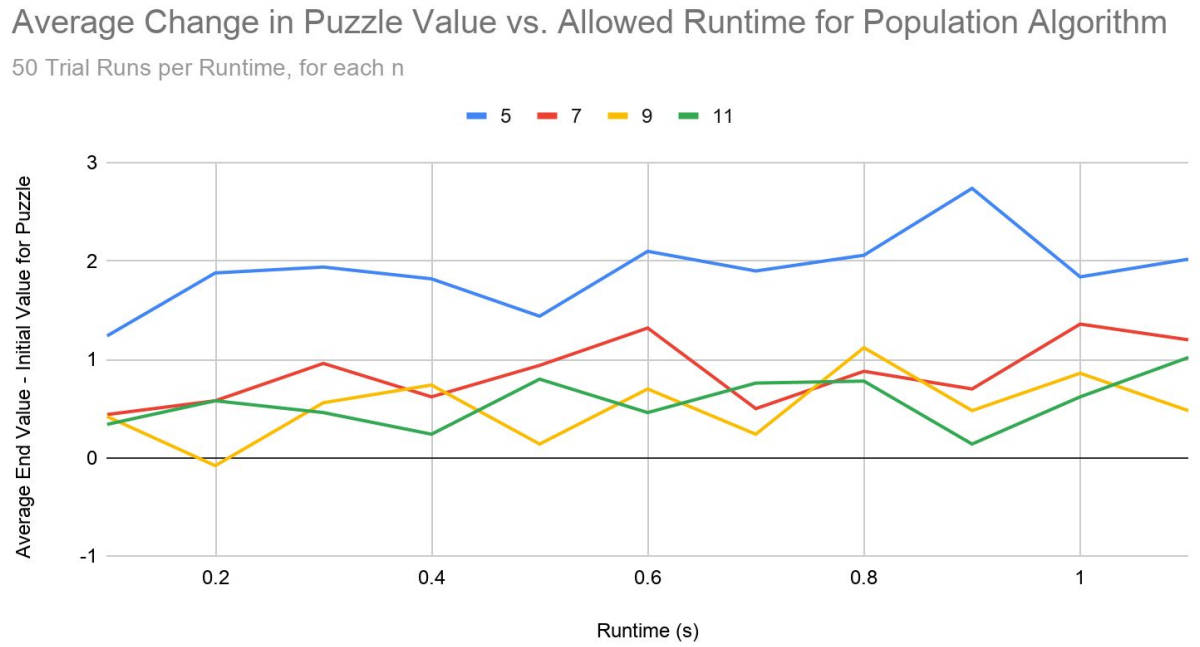
Provide a plot of how the evaluation function changes as computation time increases averaged over multiple runs of the approaches (at least 50). Report these statistics for different sizes of puzzles (n = 5, 7, 9, 11).

As before, the data used to generate all graphs is attached in the spreadsheet.

With Runtime range from 0.01 to 0.10 seconds:

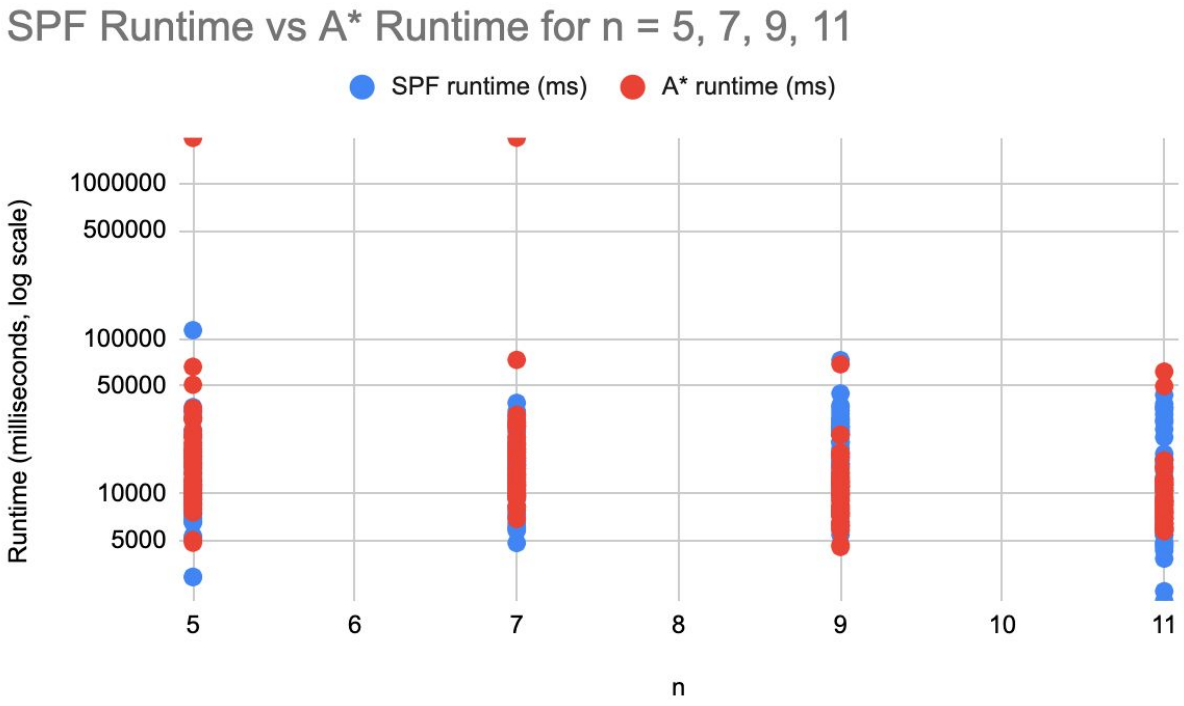


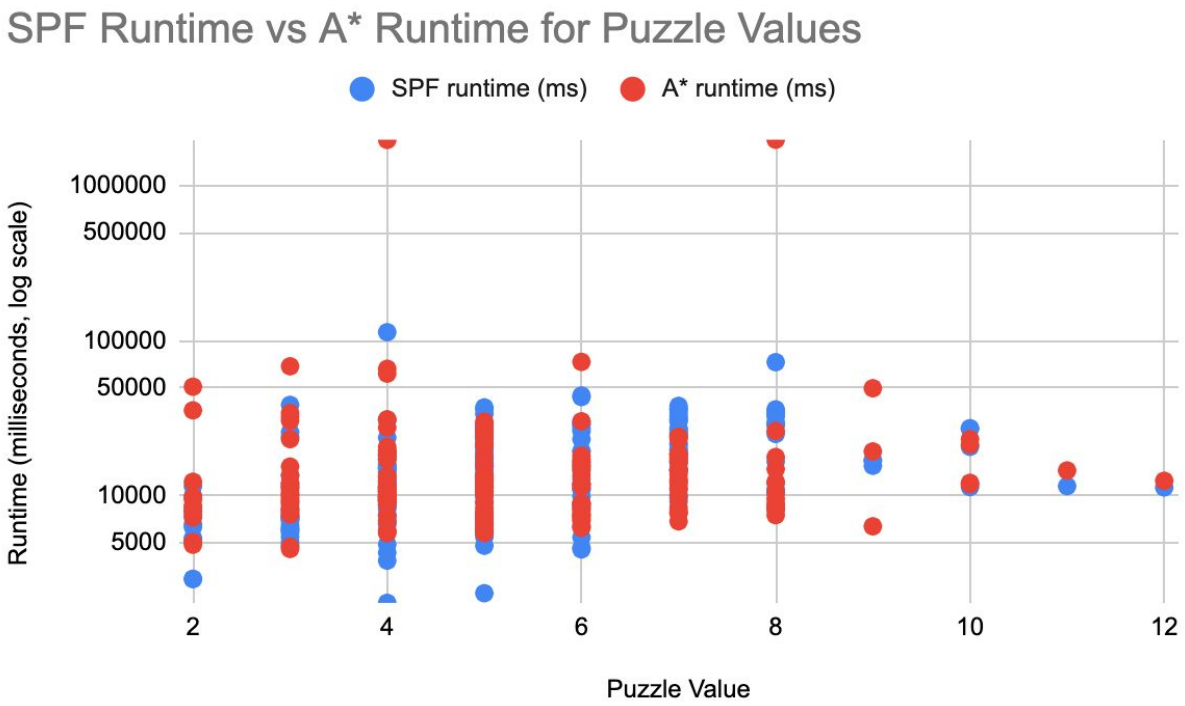
With Runtime range from 0.1 to 1.0 seconds:



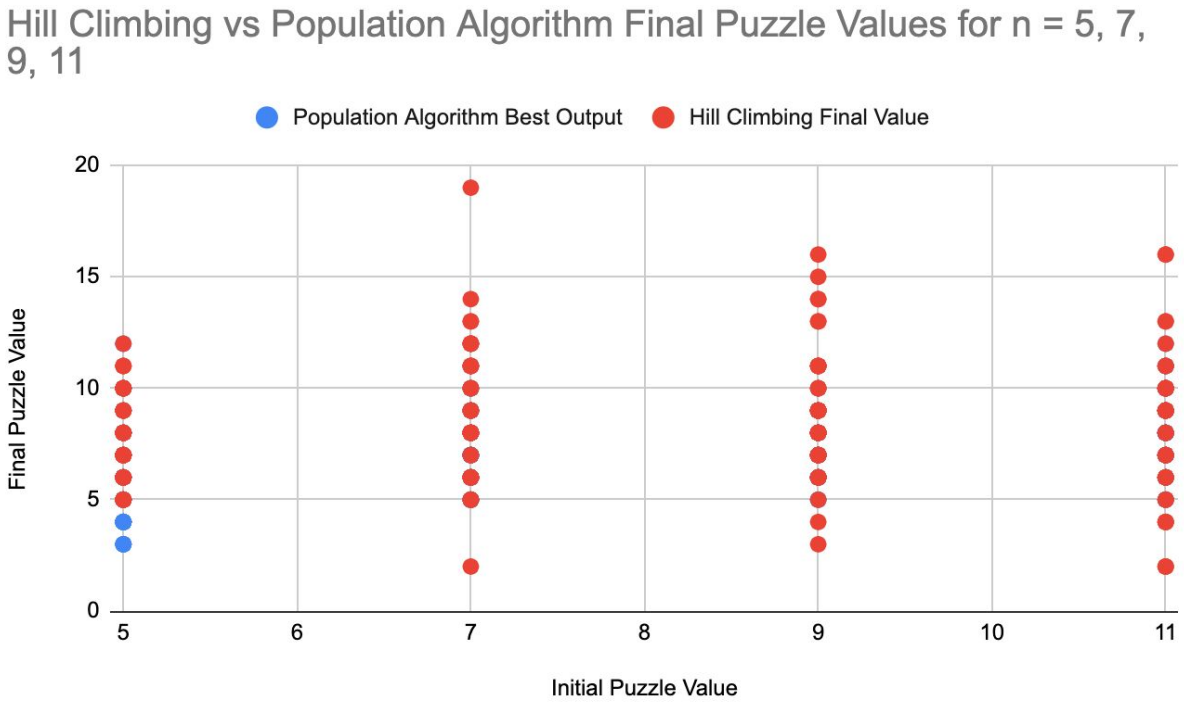
Task 8: Evaluation

To compare SPF and A*, we ran 50 trials for each size of n, with puzzles generated of various difficulty. All the data is attached in a spreadsheet. It seems like SPF is generally advantageous for most n. For more difficult puzzles however, it seems A* can edge out, perhaps since the heuristic helps to optimize which branches are explored first.

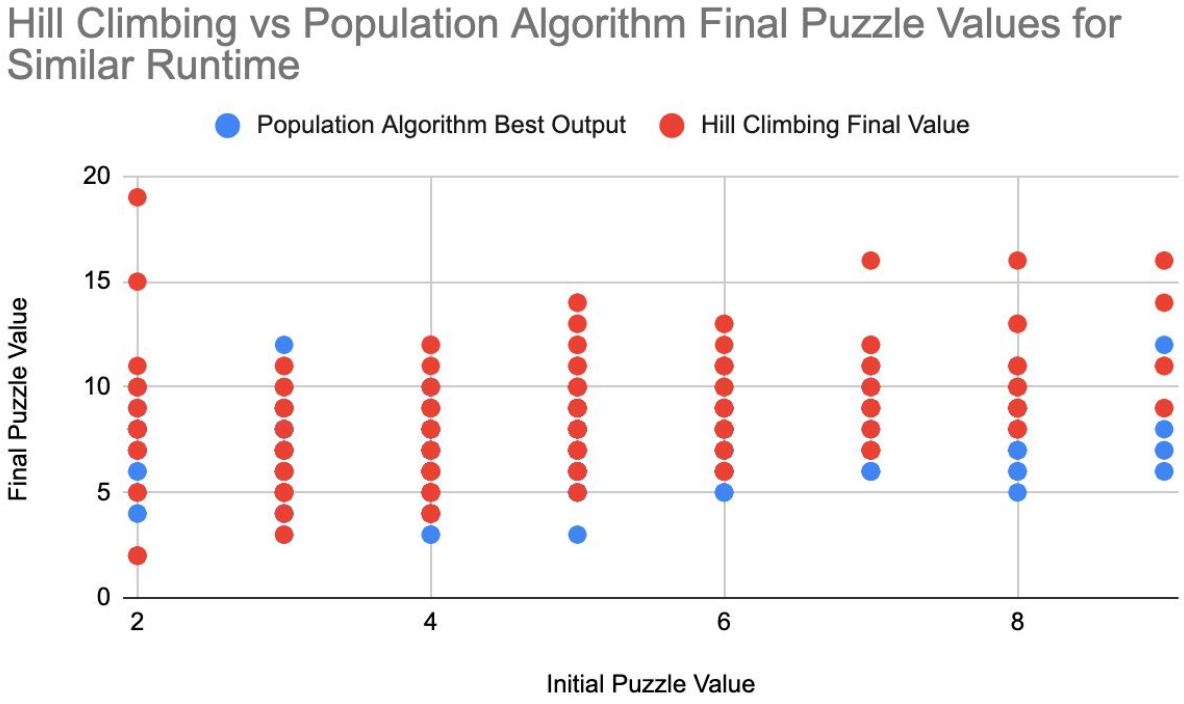




To test Hill Climbing and the Population Algorithm, Hill Climbing was run on a puzzle of size n for 100 iterations, and a random board was generated from the population algorithm for the same runtime. It is difficult to directly compare the two algorithms since the Population algorithm does not take a pre-existing board as an input, and instead begins with a randomly generated initial population, but this approximates the hardest board the Population algorithm could generate in the same time it took to run Hill Climbing for 100 iterations.



For this plot, we can see the Final Value generated by Hill Climbing when given an initial puzzle, and the output for the population algorithm given in the same time it took to generate that final puzzle.



Task 9: Extra Credit

Our code is limited only by runtime, so the largest size is technically unlimited.

However, after running the code multiple times, we generated a puzzle of size 100 with a difficulty of 13.

The puzzle and optimal path are attached in the file called ExtraCredit gameboard.rtf.