MAvis 2 – Heuristics

Nic Colvin, Justin Zheng, Kaiya Magnuson, Kevin Shah

Group Declaration

Nic: Joined partway through assignment but helped with discussions, ideas, and research

Justin: Helped with research, discussions, ideas, and coding

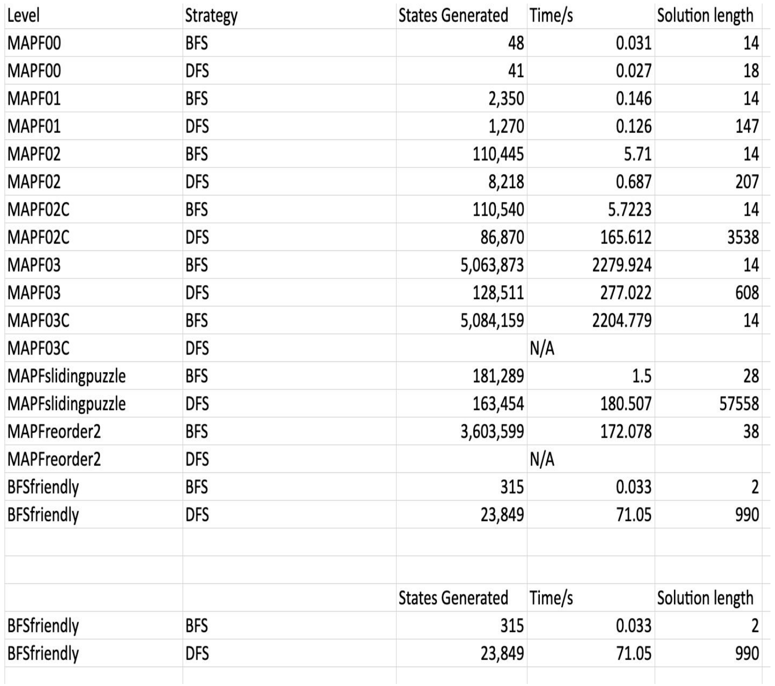
Kaiya: Coded the Manhattan distance heuristic and helped with other coding

Kevin: Helped with research, discussions, ideas, and coding. Compiled and edited video.

Implementing Best-First Search

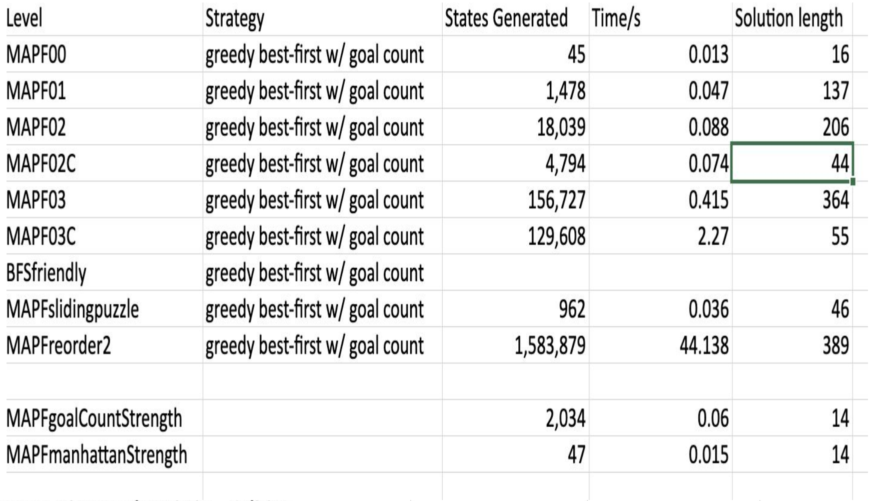
- Utilized Priority Queue and Set
- When we run A* with a heuristic that always returns 0, every node on the frontier is expanded, so A* acts like a BFS search

Comparing Benchmarks from Mavis 1

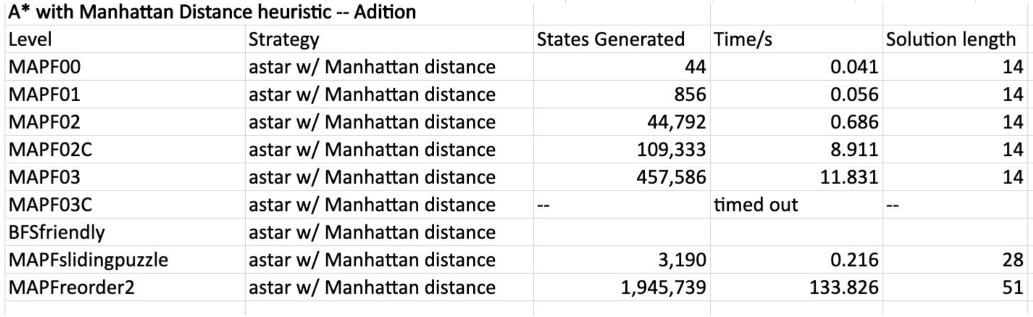


Level	Strategy	States Generated	Time/s	Solution length
MAPF00	BFS	48	0.031	14
MAPF00	DFS	41	0.027	18
MAPF01	BFS	2,350	0.146	14
MAPF01	DFS	1,270	0.126	147
MAPF02	BFS	110,445	5.71	14
MAPF02	DFS	8,218	0.687	207
MAPF02C	BFS	110,540	5.7223	14
MAPF02C	DFS	86,870	165.612	3538
MAPF03	BFS	5,063,873	2279.924	14
MAPF03	DFS	128,511	277.022	608
MAPF03C	BFS	5,084,159	2204.779	14
MAPF03C	DFS		N/A	
MAPFslidingpuzzle	BFS	181,289	1.5	28
MAPFslidingpuzzle	DFS	163,454	180.507	57558
MAPFreorder2	BFS	3,603,599	172.078	38
MAPFreorder2	DFS		N/A	
BFSfriendly	BFS	315	0.033	2
BFSfriendly	DFS	23,849	71.05	990

		States Generated	Time/s	Solution length
BFSfriendly	BFS	315	0.033	2
BFSfriendly	DFS	23,849	71.05	990



Level	Strategy	States Generated	Time/s	Solution length
MAPF00	greedy best-first w/ goal count	45	0.013	16
MAPF01	greedy best-first w/ goal count	1,478	0.047	137
MAPF02	greedy best-first w/ goal count	18,039	0.088	206
MAPF02C	greedy best-first w/ goal count	4,794	0.074	44
MAPF03	greedy best-first w/ goal count	156,727	0.415	364
MAPF03C	greedy best-first w/ goal count	129,608	2.27	55
BFSfriendly	greedy best-first w/ goal count			
MAPFslidingpuzzle	greedy best-first w/ goal count	962	0.036	46
MAPFreorder2	greedy best-first w/ goal count	1,583,879	44.138	389
MAPFgoalCountStrength		2,034	0.06	14
MAPFmanhattanStrength		47	0.015	14

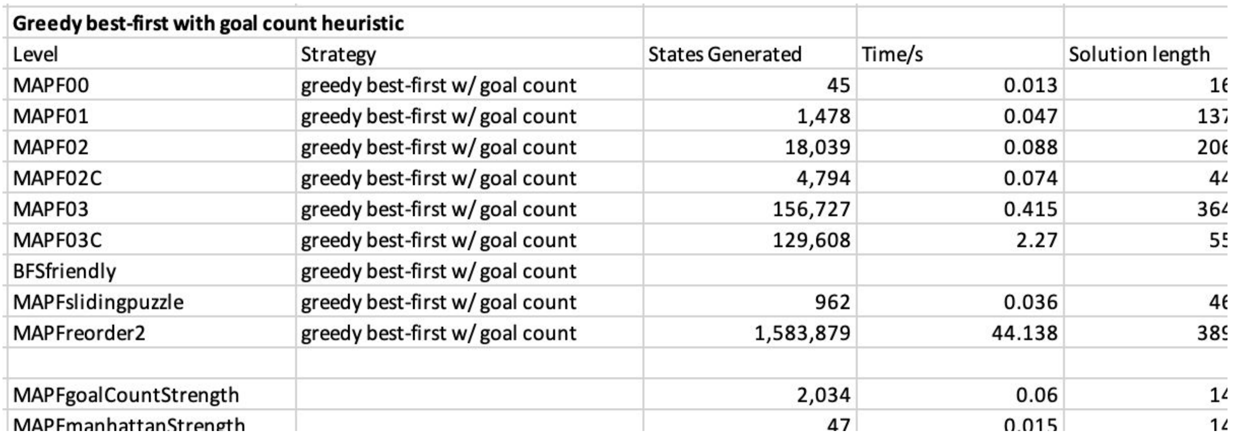


A* with Manhattan Distance heuristic -- Adition

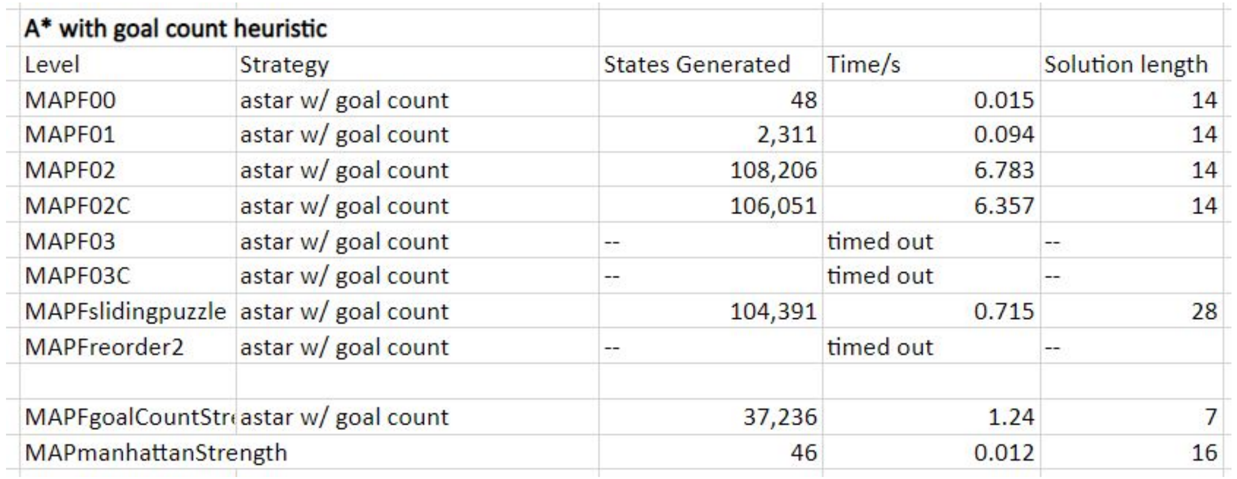
Level	Strategy	States Generated	Time/s	Solution length
MAPF00	astar w/ Manhattan distance	44	0.041	14
MAPF01	astar w/ Manhattan distance	856	0.056	14
MAPF02	astar w/ Manhattan distance	44,792	0.686	14
MAPF02C	astar w/ Manhattan distance	109,333	8.911	14
MAPF03	astar w/ Manhattan distance	457,586	11.831	14
MAPF03C	astar w/ Manhattan distance	--	timed out	--
BFSfriendly	astar w/ Manhattan distance			
MAPFslidingpuzzle	astar w/ Manhattan distance	3,190	0.216	28
MAPFreorder2	astar w/ Manhattan distance	1,945,739	133.826	51

Goal Count Heuristic

- getGoalCount():
- Goal count iterates through all agents and compares their positions to goals position to check if they have reached the correct goal
- Each call to the method returns how many empty goals are left



Greedy best-first with goal count heuristic				
Level	Strategy	States Generated	Time/s	Solution length
MAPF00	greedy best-first w/ goal count	45	0.013	16
MAPF01	greedy best-first w/ goal count	1,478	0.047	137
MAPF02	greedy best-first w/ goal count	18,039	0.088	206
MAPF02C	greedy best-first w/ goal count	4,794	0.074	44
MAPF03	greedy best-first w/ goal count	156,727	0.415	364
MAPF03C	greedy best-first w/ goal count	129,608	2.27	55
BFSfriendly	greedy best-first w/ goal count			
MAPFslidingpuzzle	greedy best-first w/ goal count	962	0.036	46
MAPFreorder2	greedy best-first w/ goal count	1,583,879	44.138	389
MAPFgoalCountStrength		2,034	0.06	14
MAPFmanhattanStrength		47	0.015	14

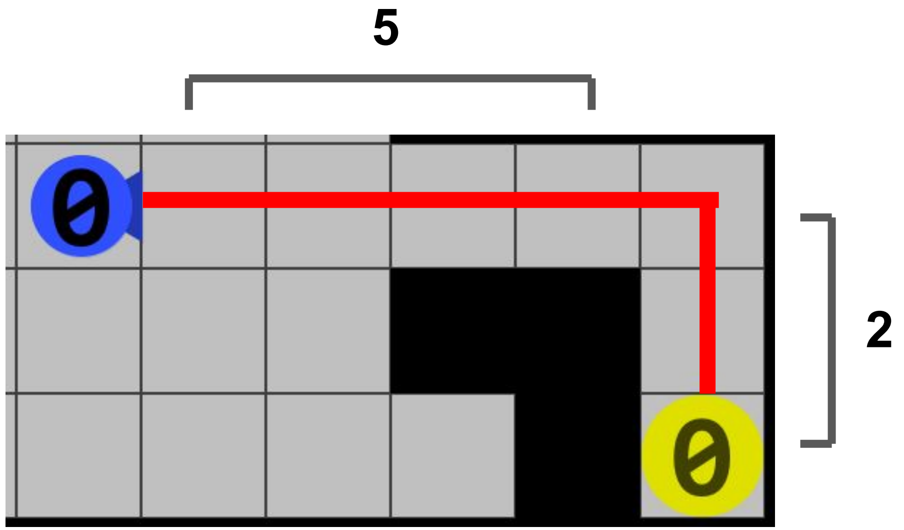


A* with goal count heuristic				
Level	Strategy	States Generated	Time/s	Solution length
MAPF00	astar w/ goal count	48	0.015	14
MAPF01	astar w/ goal count	2,311	0.094	14
MAPF02	astar w/ goal count	108,206	6.783	14
MAPF02C	astar w/ goal count	106,051	6.357	14
MAPF03	astar w/ goal count	--	timed out	--
MAPF03C	astar w/ goal count	--	timed out	--
MAPFslidingpuzzle	astar w/ goal count	104,391	0.715	28
MAPFreorder2	astar w/ goal count	--	timed out	--
MAPFgoalCountStr	astar w/ goal count	37,236	1.24	7
MAPmanhattanStrength		46	0.012	16

Manhattan Distance

- Sum of horizontal and vertical difference between the agent and the goal
- Good choice for a grid environment

Manhattan Distance: 5 + 2 = 7



Manhattan Distance Pseudocode

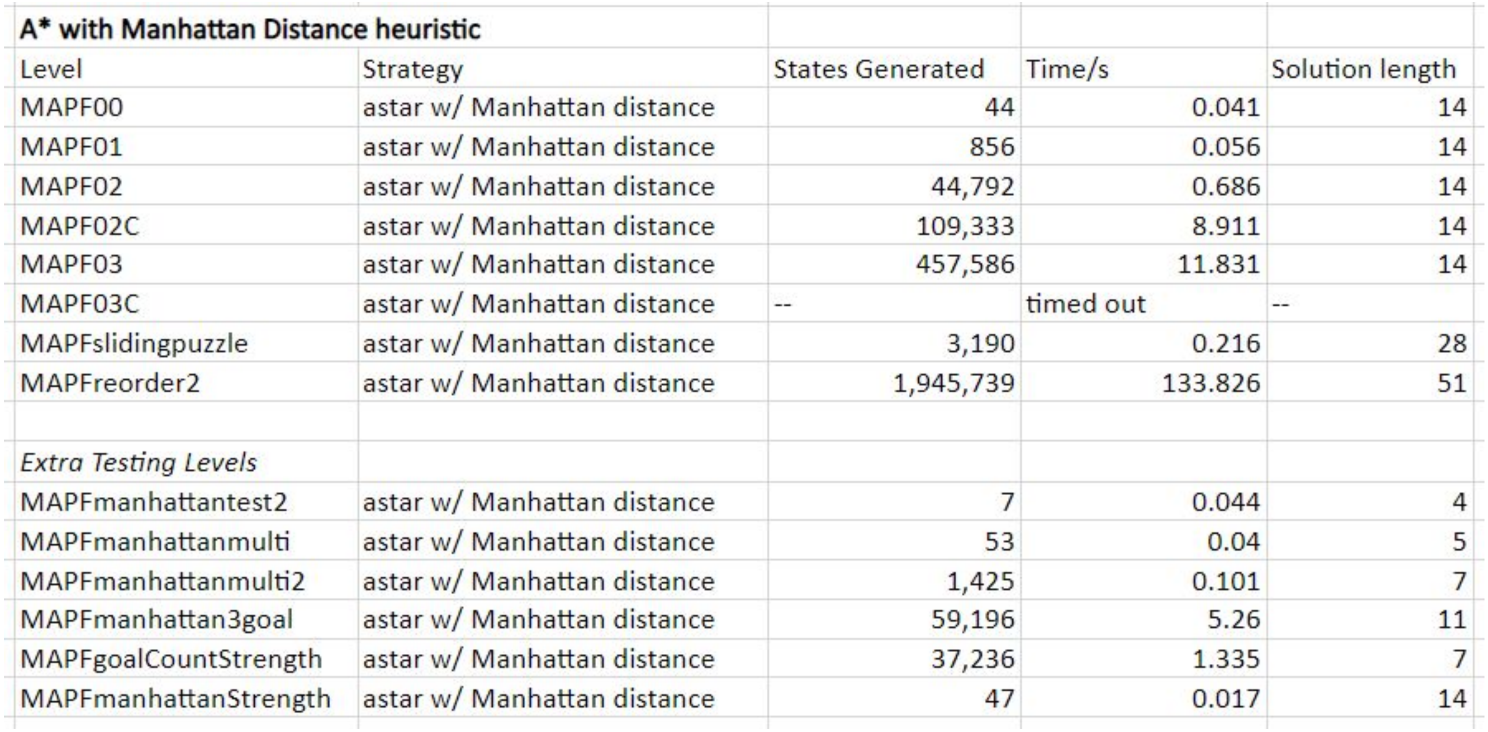
Preprocessing:

```
for (goal in goals) {
    for (x < levelLength) {
        for (y < levelHeight) {
            distance = Math.abs(goalX - x) + Math.abs(goalY - y)
            distances[x][y] = distance
        }
    }
    manhattanDistances[goal] = distances
}
```

Heuristic Function:

```
for (goal in goals) {
    sumManhattanDist += manhattanDistances[goal][agentX][agentY]
}
```

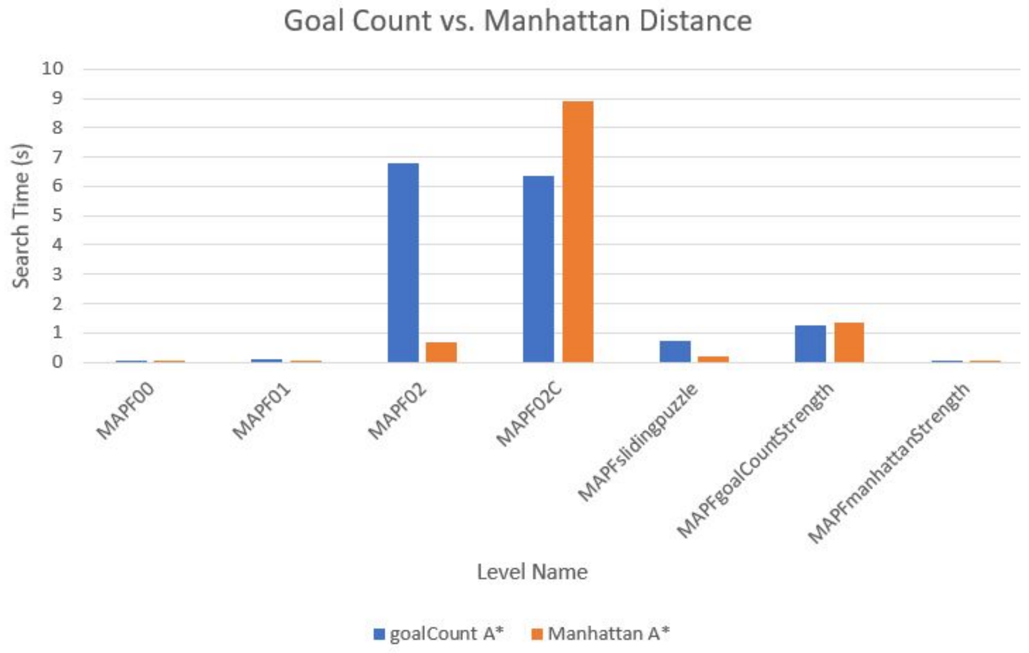
Benchmark Evaluation



A* with Manhattan Distance heuristic				
Level	Strategy	States Generated	Time/s	Solution length
MAPF00	astar w/ Manhattan distance	44	0.041	14
MAPF01	astar w/ Manhattan distance	856	0.056	14
MAPF02	astar w/ Manhattan distance	44,792	0.686	14
MAPF02C	astar w/ Manhattan distance	109,333	8.911	14
MAPF03	astar w/ Manhattan distance	457,586	11.831	14
MAPF03C	astar w/ Manhattan distance	--	timed out	--
MAPFslidingpuzzle	astar w/ Manhattan distance	3,190	0.216	28
MAPFreorder2	astar w/ Manhattan distance	1,945,739	133.826	51
Extra Testing Levels				
MAPFmanhattantest2	astar w/ Manhattan distance	7	0.044	4
MAPFmanhattanmulti	astar w/ Manhattan distance	53	0.04	5
MAPFmanhattanmulti2	astar w/ Manhattan distance	1,425	0.101	7
MAPFmanhattan3goal	astar w/ Manhattan distance	59,196	5.26	11
MAPFgoalCountStrength	astar w/ Manhattan distance	37,236	1.335	7
MAPFmanhattanStrength	astar w/ Manhattan distance	47	0.017	14

Benchmark Evaluation

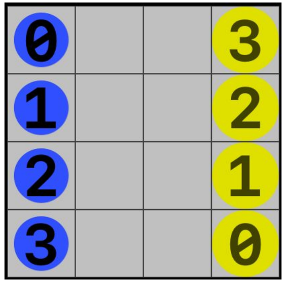
- Similar search times for simple levels
- Manhattan distance was 9.88 times faster in MAPF02
- Goal count was 1.40 times faster in MAPF02C
- Only the Manhattan distance heuristic solved MAPF03 and MAPFreorder2



Expected Goal Count vs. Manhattan Distance Strengths

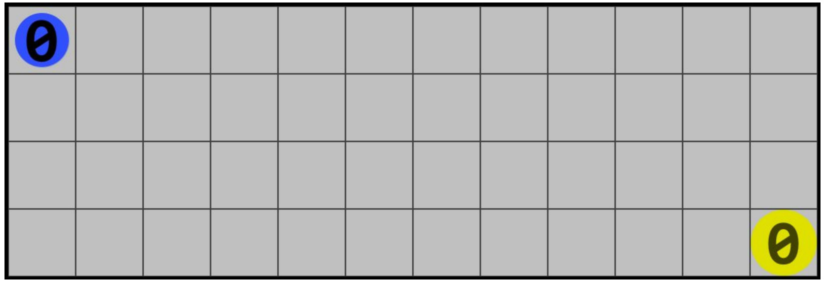
Goal Count:

- Levels with more goals to cover
- Levels where the agent is close to the goal / smaller-sized levels



Manhattan Distance:

- Levels with one / few goals
- Levels where the agent starts far from the goal



Overall Evaluation of Manhattan Distance

Weaknesses:

- No significant performance improvement over goal count
- More expensive to calculate every time a node is expanded
- Did not clearly outperform goal count in spatially larger levels, neither did goal count outperform Manhattan distance in levels with more goals

Areas for improvement:

- Experiment with other level characteristics to identify Manhattan distance strengths
- Try another heuristic

Variations on Manhattan Distance

Multiplication: `sumManhattanDist *= agent distance`

- Lone agents farther away from the goal are are weighted more than agents very close to the goal
- Similar performance to summing Manhattan distance
- 3.66 times faster for MAPF02C, which suggests it performs better on multi-agent levels

Exponentiation: $\texttt{sumManhattanDist} = \texttt{sumManhattanDist}^{\texttt{agent distance}}$

- Weights longer distances very heavily in multi-agent levels
- sumManhattanDist originally set to 1.1
- Slightly longer solve times, and timed out on the more complicated levels
- Calculation order matters!

However– admissible heuristics should **never overestimate** the cost of reaching the goal!