MAVIS 3

Kevin Shah, Kaiya Magnuson, Nic Colvin, Justin Zheng

Group Declaration

- Kevin: Assisted with implementing the push/pull code. Also contributed to designing the goal count heuristic.
- Nic: Assisted with debugging and problem solving the different heuristics and organizing/editing the video submission
- Kaiya: Updated Manhattan distance heuristic code, implemented Pull actions, ran benchmarks, assisted with Push actions, goal count code, and debugging.
- Justin: Assisted with push/pull code and goal count heuristic/code

Push and Pull Implementation

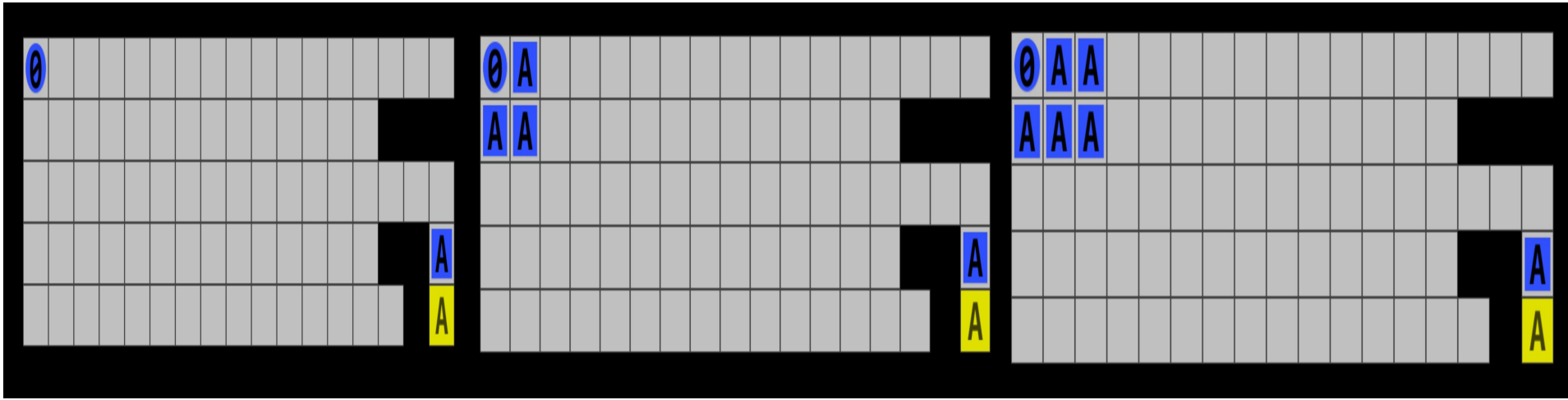
```
PullNN("Pull(N,N)", ActionType.Pull, -1, 0, -1, 0),
PullNE("Pull(N,E)", ActionType.Pull, -1, 0, 0, 1),
PullNW("Pull(N,W)", ActionType.Pull, -1, 0, 0, -1),
PullSS("Pull(S,S)", ActionType.Pull, 1, 0, 1, 0),
PullSE("Pull(S,E)", ActionType.Pull, 1, 0, 0, 1),
PullSW("Pull(S,W)", ActionType.Pull, 1, 0, 0, -1),
PullEE("Pull(E,E)", ActionType.Pull, 0, 1, 0, 1),
PullEN("Pull(E,N)", ActionType.Pull, 0, 1, -1, 0),
PullES("Pull(E,S)", ActionType.Pull, 0, 1, 1, 0),
PullWW("Pull(W,W)", ActionType.Pull, 0, -1, 0, -1),
PullWN("Pull(W,N)", ActionType.Pull, 0, -1, -1, 0),
PullWS("Pull(W,S)", ActionType.Pull, 0, -1, 1, 0),


PushNN("Push(N,N)", ActionType.Push, -1, 0, -1, 0),
PushNE("Push(N,E)", ActionType.Push, -1, 0, 0, 1),
PushNW("Push(N,W)", ActionType.Push, -1, 0, 0, -1),
PushSS("Push(S,S)", ActionType.Push, 1, 0, 1, 0),
PushSE("Push(S,E)", ActionType.Push, 1, 0, 0, 1),
PushSW("Push(S,W)", ActionType.Push, 1, 0, 0, -1),
PushEE("Push(E,E)", ActionType.Push, 0, 1, 0, 1),
PushEN("Push(E,N)", ActionType.Push, 0, 1, -1, 0),
PushES("Push(E,S)", ActionType.Push, 0, 1, 1, 0),
PushWW("Push(W,W)", ActionType.Push, 0, -1, 0, -1),
PushWN("Push(W,N)", ActionType.Push, 0, -1, -1, 0),
PushWS("Push(W,S)", ActionType.Push, 0, -1, 1, 0);
```

```
case Pull:
    // Get box's current position and ID
    int boxCurrentRow = agentRow - action.boxRowDelta;
    int boxCurrentCol = agentCol - action.boxColDelta;
    char box = this.boxes[boxCurrentRow][boxCurrentCol];

    // Update agent's position
    this.agentRows[agent] += action.agentRowDelta;
    this.agentCols[agent] += action.agentColDelta;

    // Get box's destination position
    int boxDestinationRow = agentRow;
    int boxDestinationCol = agentCol;

    // Update box's position
    this.boxes[boxCurrentRow][boxCurrentCol] = '\0';
    this.boxes[boxDestinationRow][boxDestinationCol] = box;
```

```
case Push:
    // Get box's current position and ID
    int boxCurrentRow2 = agentRow + action.agentRowDelta;
    int boxCurrentCol2 = agentCol + action.agentColDelta;
    box = this.boxes[boxCurrentRow2][boxCurrentCol2];

    // Update agent's position
    this.agentRows[agent] += action.agentRowDelta;
    this.agentCols[agent] += action.agentColDelta;

    // Get box's destination position
    int boxDestinationRow2 = boxCurrentRow2 + action.boxRowDelta;
    int boxDestinationCol2 = boxCurrentCol2 + action.boxColDelta;

    // Update box's position
    this.boxes[boxCurrentRow2][boxCurrentCol2] = '\0';
    this.boxes[boxDestinationRow2][boxDestinationCol2] = box;

    break;
```

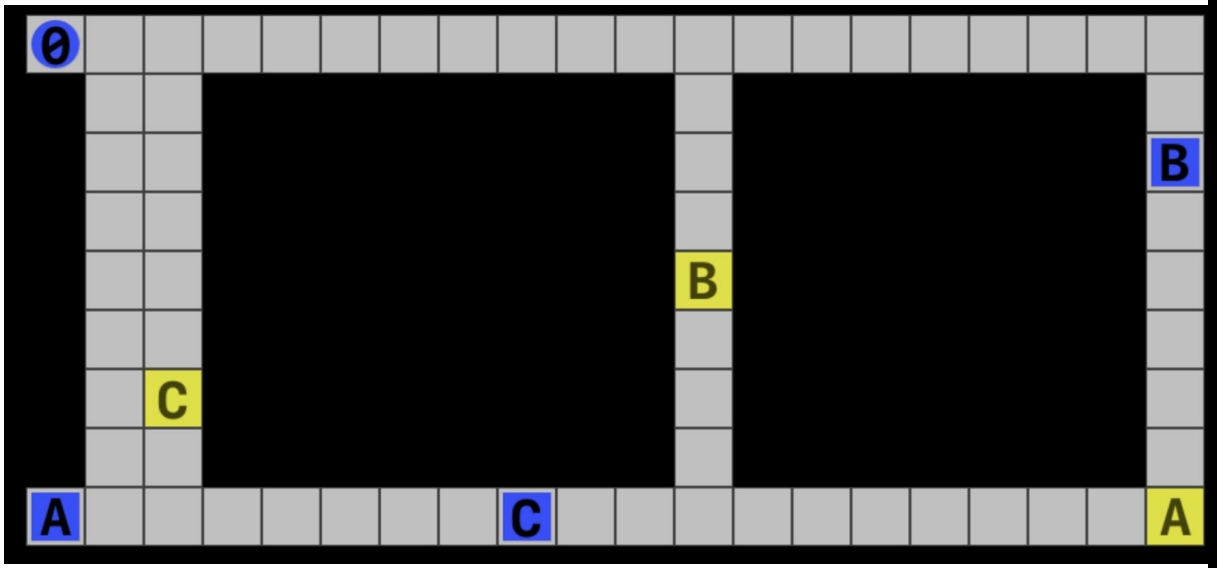
BFS on SAD Levels 1-3

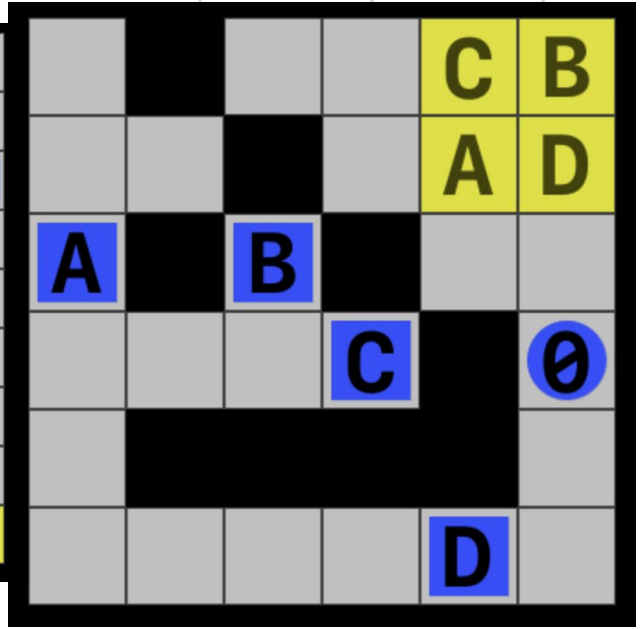
SAD1	BFS	78	0.077	19
SAD2	BFS	662,846	5.862	19
SAD3	BFS		timed out	



Push Pull Performance

BFS				DFS			
Level	States Generated	Time/s	Solution length	Level	States Generated	Time/s	Solution length
SAFirefly	1,930,781	14.291	60	SAFirefly	281,753	13.936	183250
SACrunch		timed out		SACrunch	6,469,884	27.41	1297945



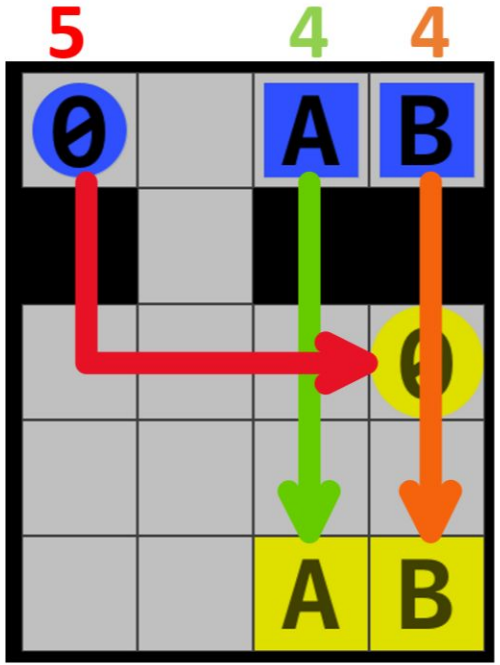


Updating Goal Count Heuristic for Single-Agent Levels with Boxes

- Calculating whether a box is in a goal as opposed to the agent being in the goal
- Uses same logic to find where the goals are
- Must make sure that goals have the correct box in them to calculate goal count

```
// Calculates the number of goals with a correct box – Exercise 6.1
for (int i = 0; i < numGoals; i++) {
    int goalX = goalCoords[i][1];
    int goalY = goalCoords[i][0];

    if (goals[goalX][goalY] != '\0') {
        if (goals[goalX][goalY] == boxes[goalX][goalY]) {
            //System.err.println(" Counted box goals[" + goalX +"][" + goalY + "] = " + goals[goalX][goalY] + ".");
            //System.err.println(this);

        public int getGoalCount() {

            //System.err.println("----Counting goals----");
            //System.err.println(this);

            // agentRows is an array of each agent's row position (agents identified numerically)
            int numAgents = agentRows.length;
            int goalCount = 0;

            // // Calculates the number of agents at their correct goal
            for (int agentNum = 0; agentNum < numAgents; agentNum++) {
                // If selected goal has same row and col as agent
                int agentX = agentRows[agentNum];
                int agentY = agentCols[agentNum];

                // Get numerical goal ID by converting from ASCII
                int goalNum = (int) goals[agentX][agentY] - 48;

                if (goalNum == agentNum) {
                    System.err.println("Counted agent goal " + agentNum);
                    goalCount++;
                }
            }

            // Calculates the number of goals with a correct box – Exercise 6.1
            for (int i = 0; i < numGoals; i++) {
                int goalX = goalCoords[i][1];
                int goalY = goalCoords[i][0];

                if (goals[goalX][goalY] != '\0') {
                    if (goals[goalX][goalY] == boxes[goalX][goalY]) {
                        //System.err.println(" Counted box goals[" + goalX +"][" + goalY + "] = " + goals[goalX][goalY] + ".");
                        //System.err.println(this);

                        goalCount++;
                    }
                }

            }
```

Updated Manhattan Distance Heuristic



Example: `h` **= 5 + 4 + 4 = 13**

Pseudocode:

Preprocessing:

For every goal in level:

    Generate Manhattan distance lookup table

    Store goal ID / lookup table pairing as a `DistanceGrid` obj. in `gridLookup` array

h-function:

For every `DistanceGrid` in `gridLookup`:

    If it has an agent goal ID:

```
        distance = distanceGrid.distances[agentX][agentY]
sumManhattanDistances += distance
```

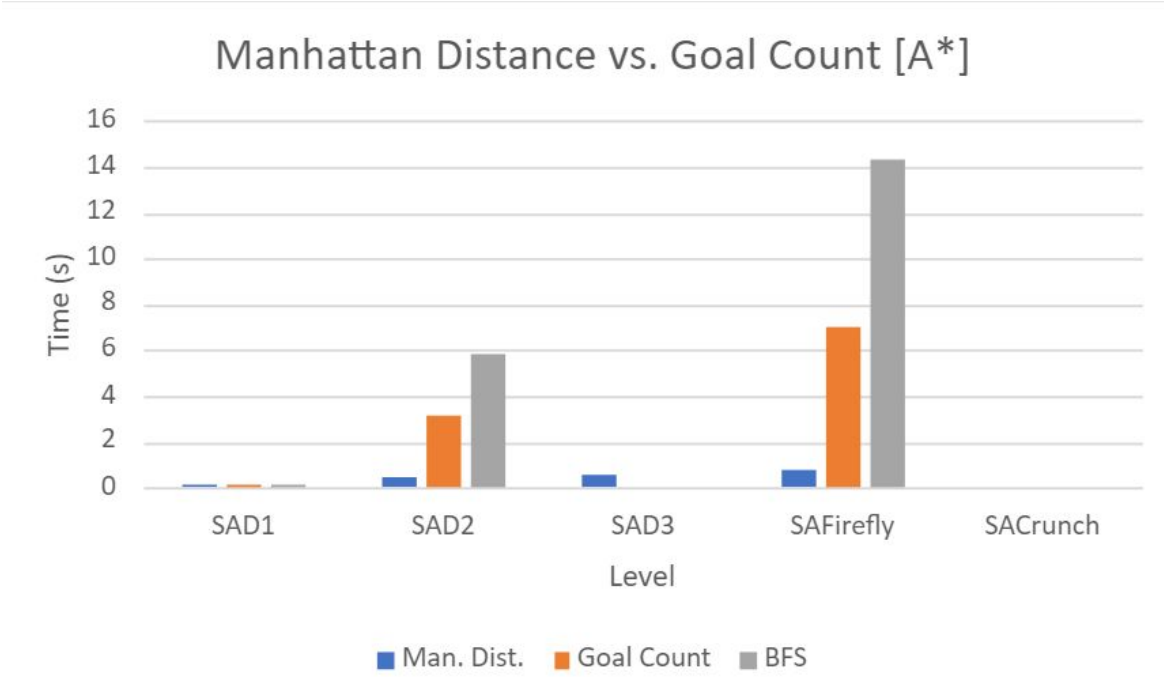
For every box in `State.boxes`:

    For every `DistanceGrid` in `gridLookup` with matching goal ID:

```
        distance = distanceGrid.distances[boxX][boxY]
        if (distance < minDistance):
            minDistance = distance
sumManhattanDistances += minDistance
```

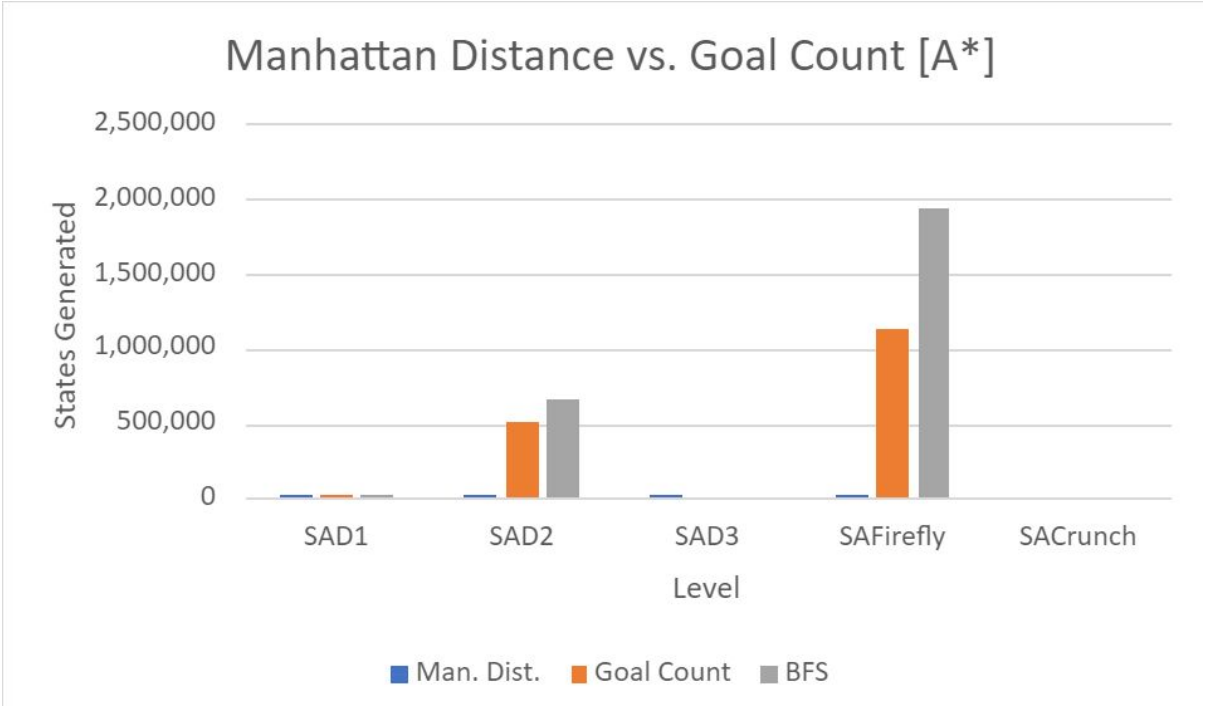
Solution Time Benchmarks

- Significantly faster than goal count & BFS
- Only method that solved SAD3 with A*
  - Handles multi-agent/box levels well

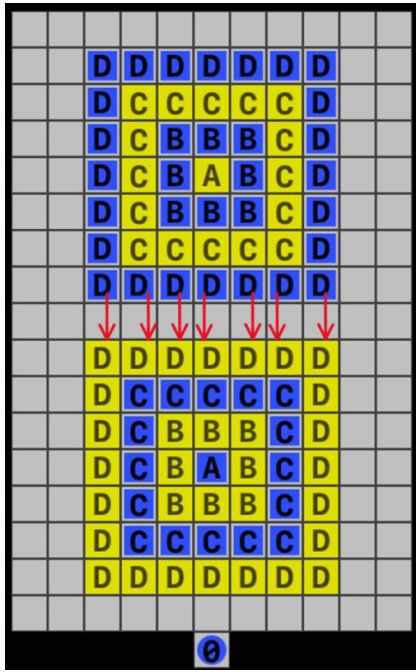


States Generated Benchmarks

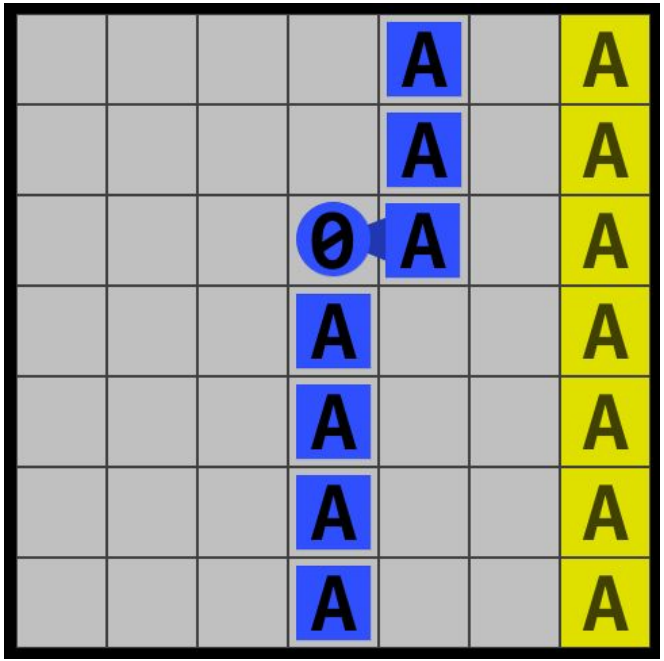
- Generates dramatically fewer states
- Expanded states are more likely to lead to the goal



Funny Behavior



SAWatsOn



SAsoko3_07

Ideal Level Types

- Performs better when agents/boxes have goals
- Struggles on levels where agent/box order matters more than proximity to goal
- Takes longer to generate states