## Vir Khanna

#### Programming Homework 1

#### ECS 170

#### Professor Davidson

#### 28 Jan 2024

The project explores the creation of admissible heuristics for the A\* algorithm in a simulated Mt. St. Helens environment. The challenge lies in tailoring heuristics to the varied terrain, with the goal of efficient and optimal pathfinding. This write-up details the process and findings from testing three A\* variants against their expected performance.

## Designing Admissible Heuristics

### AStarExp():

Chebyshev Distance: I define the Chebyshev distance (dC) as dC = max(|x\_goal - x\_current|, |y\_goal - y\_current|). This measures the minimum steps needed on flat terrain since each move costs at least 1, making it a lower bound on the actual cost.

Elevation Change: For elevation, let h\_current and h\_goal be the elevations at the current and goal nodes. I calculate the elevation component (h\_elev) only for upward movement as h\_elev = max(h\_goal - h\_current, 0).

Combined Heuristic: The total heuristic (h) is then h = dC + h\_elev. This approach ensures admissibility because dC represents the minimal flat movement cost, and h\_elev adds cost only for upward movement, aligning with the actual cost without exceeding it.

### AStarDiv():

Chebyshev Distance: I calculate the Chebyshev distance as dC = max(abs(goal.x - current.x), abs(goal.y - current.y)), which estimates the minimum number of steps required on a flat surface.

Elevation Change: The elevation change is considered between the current position (h\_current) and the goal (h\_goal). I compute this as h\_elev = max(h\_goal - h\_current, 0) but do not use it directly in the heuristic for AStarDiv.

Final Heuristic: My heuristic for AStarDiv is simply the Chebyshev distance. It's defined as h = max(dC, 0). This approach ensures admissibility because the ‘div’ cost function reduces the cost for upward movement and is at least 1 for flat or downward movement, so the heuristic never overestimates the cost.

### AStarMSH():

Chebyshev Distance: Calculated as dC = max(abs(goal.x - current.x), abs(goal.y - current.y)), it measures the minimum steps on flat terrain.

Elevation Change: Computed only for upward movement as h\_elev = max(h\_goal - h\_current, 0).

Final Heuristic: The sum h = dC + h\_elev, ensuring it never overestimates the true cost, as dC is the minimum step count and h\_elev accounts for the additional cost of climbing.

#### Benefits of Weighted A\* in This Project:

It speeds up the search, focusing more on goal-directed exploration.

In complex terrains like Mt. St. Helens, quicker solutions are valuable.

Weighted A\* balances between finding a viable path and computational efficiency.

#### Drawbacks:

It may lead to paths that are not the shortest.

Requires fine-tuning the weight to balance speed and optimality.

Not ideal when the shortest path is strictly necessary.

I considered using the weighted A\* approach in this project to efficiently handle complex terrains, prioritizing faster solutions over the absolute shortest paths, given the challenging and varied nature of the terrain.

## Testing

### AStarExp():

Seed 0 Testing:

Expected Path Cost: 241.55253710831192

Actual Path Cost: 241.55253710831192

The algorithm successfully found the optimal path, matching the expected cost exactly.

Seed 1 Testing:

Expected Path Cost: 238.5066083803129

Actual Path Cost: 238.5066083803129

The algorithm successfully found the optimal path, matching the expected cost exactly.

Seed 2 Testing:

Expected Path Cost: 236.667690049357

Actual Path Cost: 236.667690049357

The algorithm successfully found the optimal path, matching the expected cost exactly.

Seed 3 Testing:

Expected Path Cost: 422.01798243905006

Actual Path Cost: 422.01798243905006

The algorithm successfully found the optimal path, matching the expected cost exactly.

Seed 4 Testing:

Expected Path Cost: 254.34464507852198

Actual Path Cost: 254.34464507852198

The algorithm successfully found the optimal path, matching the expected cost exactly.

### AStarDiv():

Seed 0:

Expected Path Cost: 197.67690128917357

Actual Path Cost: 197.67690128917357

The algorithm successfully found the optimal path, matching the expected cost exactly.

Seed 1:

Expected Path Cost: 197.71712595331226

Actual Path Cost: 197.71712595331226

The algorithm successfully found the optimal path, matching the expected cost exactly.

Seed 2:

Expected Path Cost: 197.58110211547168

Actual Path Cost: 197.58110211547168

The algorithm successfully found the optimal path, matching the expected cost exactly.

Seed 3:

Expected Path Cost: 196.29558632629682

Actual Path Cost: 196.29558632629682

The algorithm successfully found the optimal path, matching the expected cost exactly.

Seed 4:

Expected Path Cost: 197.24833470274933

Actual Path Cost: 197.24833470274933

The algorithm successfully found the optimal path, matching the expected cost exactly.

### AStarMSH():

Expected Path Cost: Approximately 515.81

Actual Path Cost: 516.09

The algorithm found a path, matching within 0.3 of the optimal path.