ECE276B-HW2

Zhenyu Wu | PID: A69030822

May 11, 2025

1 Q1

1.1 Q1.1

Based on the $G = (V, E)$, we have vertices $V = \{S, A, B, C, D, E, F, T\}$ where T is the goal node, and we have $|V| = 8$ elements. For the DSP problem, we have the planning horizon $T = |V| - 1 = 8 - 1 = 7$. To perform DP, form tuple (t, v) where $t \in T$, $v \in V$, and we can express value and policy as $Value(t, v)$, $\pi(t, v)$.

1.2 Q1.2

To apply the backward DP, we start from the goal node $\tau = T$. We initialize following value and policy:

1. $V(T, \tau) = V(T-1, \tau).. = V(0, \tau) = 0$
2. $V(T, i) = \infty, \forall i \in V \setminus \{\tau\}$
3. $V(T-1, i) = c_{i,\tau}, \forall i \in V \setminus \{\tau\}$
4. $\pi(T-1, i) = \tau, \forall i \in V \setminus \{\tau\}$

Then we compute

1. $V(t, i) = \min_{j \in V}(c_{i,j} + V(t+1, j)), \forall i \in V \setminus \{\tau\}$
2. $\pi(t, i) = argmin_{j \in V}(c_{i,j} + V(t+1, j)), \forall i \in V \setminus \{\tau\}$

from $t = T - 2$ to 0. The answer would be find at $V(0, i)$ which is the optimal cost-to-go from node i to τ in at most T step. Below is my results.

```
Start from A: ['D', 'T'], cost:48
Start from B: ['F', 'E', 'T'], cost:38
Start from C: ['F', 'E', 'T'], cost:45
Start from D: ['T'], cost:27
Start from E: ['T'], cost:16
Start from F: ['E', 'T'], cost:30
Start from S: ['C', 'F', 'E', 'T'], cost:55
```

Figure 1: Q1.2

2 Q2

Since we have a symmetric robot (line segment), we use Minkowski sum to compute the C_{obs} in C-space. To compute Minkowski sum, we can sample some points from robot (line segment) and the boundary of the obstacle (circle) and perform $R \bigoplus O = \{a + b | a \in R, b \in O\}$ where R is the robot and O is the obstacle. For the better visualization, I apply the shapely library to compute Minkowski sum. Below is the result when $\theta = 0$.

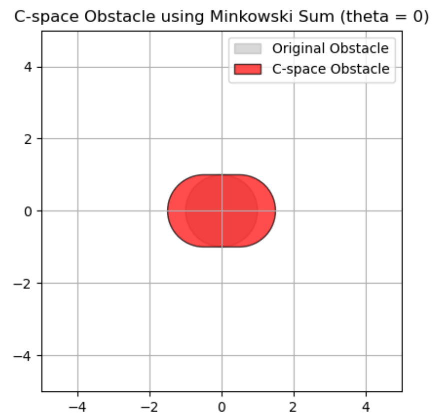


Figure 2: C space at $\theta = 0$

To handle different orientation, first we can express the front tip and back tip of the robot as $p_f = (0.5, 0)$, $p_b = (-0.5, 0)$. Then we apply rotation on them with rotation matrix $R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$. Then I have following C-space under different orientation

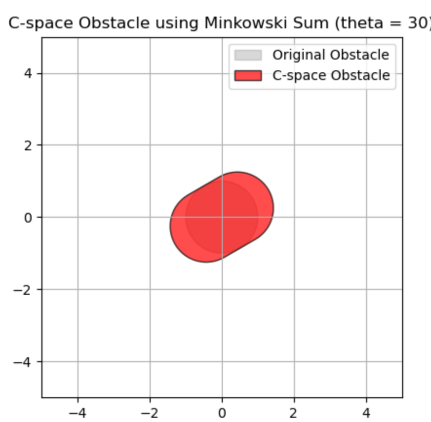


Figure 3: C space at $\theta = 30$

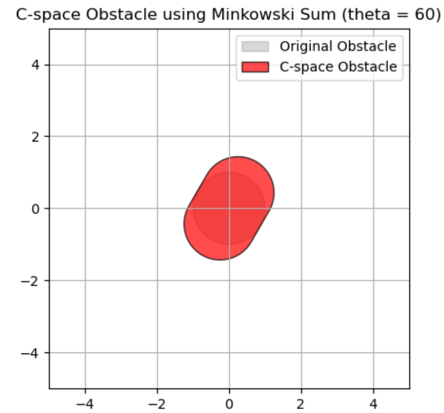


Figure 4: C space at $\theta = 60$

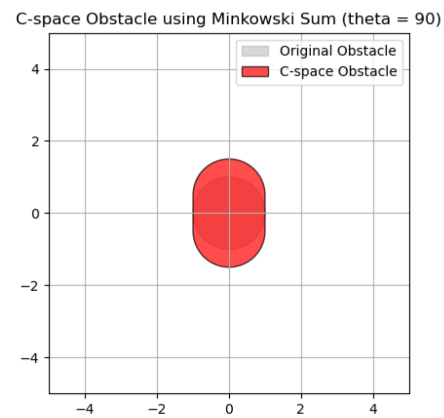


Figure 5: C space at $\theta = 90$

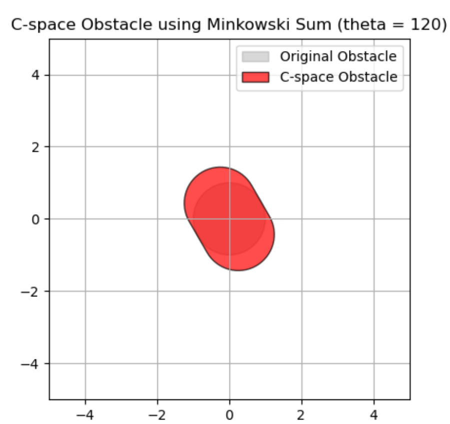


Figure 6: C space at $\theta = 120$

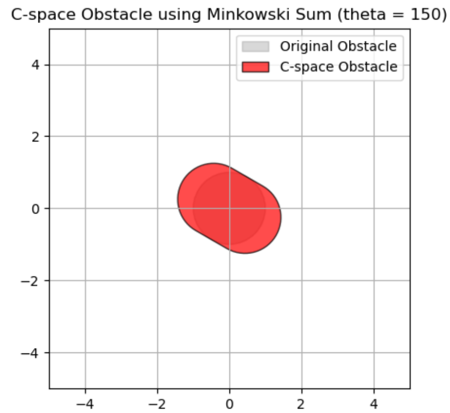


Figure 7: C space at $\theta = 150$

Lastly, to visualize C-space in 3-D view where θ is the 3-rd axis, I sample some of the θ in range $[-\pi, \pi)$, and compute Minkowski sum on each them. After stack all of them together, we have following C-space.

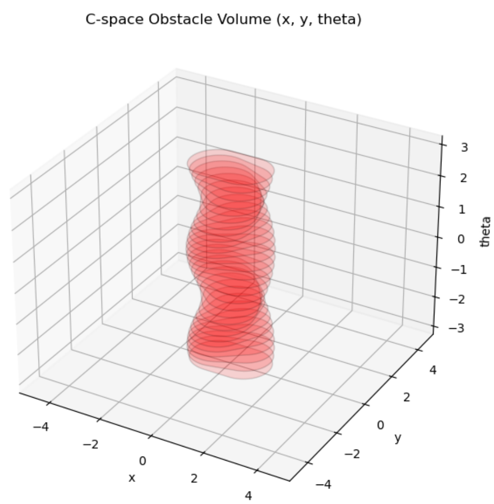


Figure 8: C space in 3D

3 Q3

3.1 Q3.1

Below is my hand-written solution

a) For goal node τ, both are consistent

$$h^{(1)}(x_\tau) = 0$$

$$h^{(2)}(x_\tau) = 0$$

$$h = \max(h^{(1)}(x_\tau), h^{(2)}(x_\tau)) = 0$$

For every other node, we have

$$h^{(1)}(x_i) \leq c_{ij} + h^{(1)}(x_j) \quad (1)$$

$$h^{(2)}(x_i) \leq c_{ij} + h^{(2)}(x_j)$$

Lets denote $h(x_j)$ as

$$h(x_j) = \max(h^{(1)}(x_j), h^{(2)}(x_j))$$

$$\therefore h^{(1)}(x_j) \leq h(x_j), \quad h^{(2)}(x_j) \leq h(x_j)$$

Substitude back to (1), we have

$$h^{(1)}(x_i) \leq c_{ij} + h(x_j) \qquad h^{(2)}(x_i) \leq c_{ij} + h(x_j)$$

$$\therefore \max(h^{(1)}(x_i), h^{(2)}(x_i)) \leq c_{ij} + h(x_j)$$

$$\therefore h(x_i) \leq c_{ij} + h(x_j) \text{ where}$$

$$h := \max(h^{(1)}, h^{(2)})$$

$\therefore h$ is also consistent

Figure 9: Q3.1

3.2 Q3.2

Below is my hand-written solution

b) For goal node τ, we have

$h^{(1)}(x_\tau) = 0 \qquad h^{(2)}(x_\tau) = 0$

$\therefore h(x_\tau) = h^{(1)}(x_\tau) + h^{(2)}(x_\tau) = 0$

For all other nodes

$\because h^{(1)}(x_i) \leq c_{ij} + h^{(1)}(x_j)$

$h^{(2)}(x_i) \leq c_{ij} + h^{(2)}(x_j)$

$\therefore h^{(1)}(x_i) + h^{(2)}(x_i) \leq c_{ij} + h^{(1)}(x_j) + c_{ij} + h^{(2)}(x_j)$

$\therefore h^{(1)}(x_i) + h^{(2)}(x_i) \leq 2c_{ij} + (h^{(1)}(x_j) + h^{(2)}(x_j))$

$\because h(x) = h^{(1)}(x) + h^{(2)}(x)$

$\therefore h(x_i) \leq 2c_{ij} + h(x_j), \quad \varepsilon = 2 > 1$

$\therefore h$ is ε-consistent

Figure 10: Q3.2

4 Q4

To implement RTAA*, I follow the pseudo code of A* and RTAA* in the lecture slides. For data structure, I apply the priority queue in implementation. The overall workflow is

1. Use A* with current start node to expand $N = 4$ nodes
2. Find the best node j^* from the OPEN list where $j^* = argmin_{j \in OPEN}(f_j)$.
3. Update Heuristic of the expanded node $h_i = f_j - g_i, \forall i \in CLOSE$.
4. Move to j^* and record the movement.

In the instruction, it says If two nodes $i, j \in V$ have the same f-values, $f_i = f_j$, then expand the node with the smaller index. I enforced this rule in A* implementation as well as the Find the best node j^* from the OPEN list step in RTAA*. Below is my results.

4.1 Iteration 1

1. Current Position: 1
2. CLOSED list: $[1, 2, 3, 4]$
3. OPEN list and f

OPEN	**OPEN f**
5	5
6	5

4. Heuristic

i	1	2	3	4	5	6	7	8	9	10
$h(i)$	5	4	3	2	3	4	3	2	1	0

5. Move to: 5 (smaller index than 6)

4.2 Iteration 2

1. Current Position: 5
2. CLOSED list: $[5, 4, 2, 3]$
3. OPEN list and f

OPEN	**OPEN f**
1	7
6	5

4. Heuristic

i	1	2	3	4	5	6	7	8	9	10
$h(i)$	5	4	3	4	5	4	3	2	1	0

5. Move to: 6

4.3 Iteration 3

1. Current Position: 6
2. CLOSED list: $[6, 7, 8, 9]$
3. OPEN list and f

OPEN	**OPEN** f
1	6
5	6
10	4

4. Heuristic

i	1	2	3	4	5	6	7	8	9	10
$h(i)$	5	4	3	4	5	4	3	2	1	0

5. Move to: 10

4.4 Final Path

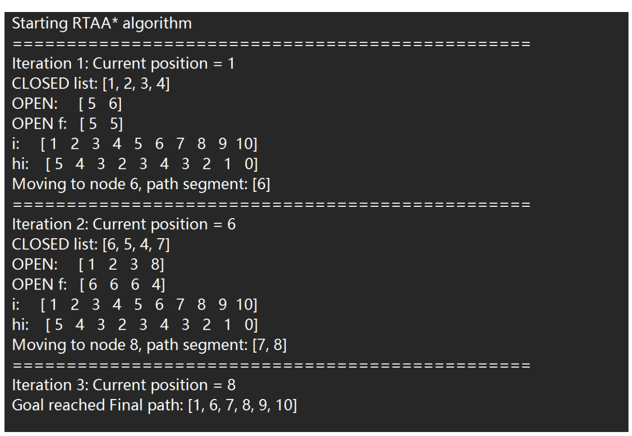
The final path from above iterations is $1 \to 2 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10$.

4.5 Additional

I'm not entirely sure whether the following rule applies only to A* expansion or if it also applies to the **"Find the best node j^* from the OPEN list"** step in RTAA*:

> If two nodes $i, j \in V$ have the same f-values, $f_i = f_j$, then expand the node with the smaller index.

The above iteration results enforced this rule in **"Find the best node j^* from the OPEN list"** step in RTAA*. Specifically, in iteration 1, it select 5 as j^* and move to 5 instead of 6 (both have same f). Therefore, I also print out another version where this rule is only applied to A*, and below is the screen shot of the results. **Please Ignore this one if the rule should be enforced in RTAA* "Find the best node j^* from the OPEN list" step**



```
Starting RTAA* algorithm
================================================
Iteration 1: Current position = 1
CLOSED list: [1, 2, 3, 4]
OPEN:    [5  6]
OPEN f:  [5  5]
i:    [1  2  3  4  5  6  7  8  9 10]
hi:   [5  4  3  2  3  4  3  2  1  0]
Moving to node 6, path segment: [6]
================================================
Iteration 2: Current position = 6
CLOSED list: [6, 5, 4, 7]
OPEN:    [1  2  3  8]
OPEN f:  [6  6  6  4]
i:    [1  2  3  4  5  6  7  8  9 10]
hi:   [5  4  3  2  3  4  3  2  1  0]
Moving to node 8, path segment: [7, 8]
================================================
Iteration 3: Current position = 8
Goal reached Final path: [1, 6, 7, 8, 9, 10]
```

Figure 11: RTAA* additional results

For this one, the path is $1 \to 6 \to 7 \to 8 \to 9 \to 10$.

5 Q5

5.1 Q5.1

I followed the weighted A* algorithm on Slide 21 of Lecture 8 to implement weighted A*. Specifically, it has following changes:

1. Use $f_i = g_i + \epsilon h_i$ as key in priority queue.
2. Keep track of v_i, set $v_i = g_i$ when node i is popped from OPEN and inserted in CLOSED.
3. If update $g_j \leftarrow g_i + c_{i,j}$ occurs on any node j that has already in CLOSED (making $v_j > g_j$), insert j in INCONS list.

Below is the table

Table 1: Weighted A* Algorithm

Iteration	**Node exiting OPEN**	**OPEN**	g_1	g_2	g_3	g_4	g_5	g_6	g_7
0	–	{2}	∞	0	∞	∞	∞	∞	∞
1	2	{5,6}	∞	0	∞	∞	9	1	∞
2	5	{3, 6}	∞	0	10	∞	9	1	∞
3	6	{1, 3, 7}	6	0	10	∞	9	1	6
4	1	{3, 7}	6	0	10	∞	8	1	6
5	3	{4, 7}	6	0	10	11	8	1	6

5.2 Q5.2

Node 5 is inconsistent node. At iteration 2, node 5 exits the OPEN list and set $v_5 = g_5 = 9$. However, later at iteration 4, when update children of node 1, it updates $g_5 = 8 < v_5 = 9$ which makes node 5 inconsistent.

References