Planning, Execution, and Learning 15-887
Homework 2

Abhishek Bhatia (abhatia1)

1) **Moving Target**

a) Planner that computes the least cost path that allows the robot to catch the moving target.

Approach: To determine the least cost path, I am using a forward dijkstra approach to determine the cost of reaching every node from the initial robot location. Then, I do a backward A* search by adding all the target locations in the priority queue initially and plan the optimal path to the initial robot location (0,0). I use the costs calculated during the forward dijkstra search as an informed heuristic for my backward A* search.

Testcase 1:
Execution Time: 0.205 msecs
Number of nodes expanded: 22
Cost of the path: 6

Testcase 2:
Execution Time: 71781.7 msecs
Number of nodes expanded: 4508876
Cost of the path: 88434
Note: These is a small bug in my code that I was not able to fix in time, hence my path cost is more than what I am getting for the non optimal planner for this test case. However, since I am using the most informed heuristic during my backward A* search, the planner should return the optimal path. The bug is most probably in how I am calculating the cost from the computed path.

b) Planner that is computationally faster, at the expense of the optimality of the solution of the path computed.

Approach: Weighted A* with goal updated at every iteration as the next target position. The idea behind this approach is to use the weighted A* algorithm to always push the robot towards the target, and since the next target location is always known, we use that information as our goal for every iteration. However, the weight that worked best for the provided test cases was ‘1’, leaving this to be the simple A* algorithm with updated goal at every iteration.

Testcase 1:
Execution Time: 0.114 msecs
Number of nodes expanded: 4
Cost of the path: 7

Testcase 2:
Execution Time: 9538.21 msecs
Number of nodes expanded: 542947
Cost of the path: 83158

Comparison of the 2 planners: As it is evident from the information above, although the planner in part a) returns an optimal path, it expands approximately 10 times more of nodes to determine the optimal path and takes approximately 7 times more time to do so. Hence, the optimal path is achieved at the expense time and space complexity. Planner in part b) will return a path in less time but there is no guarantee of optimality implying that the path identified may be of huge cost, or maybe through obstacles which may not be desirable in many scenarios. Hence, both the planners have their pros and cons and a proper analysis of requirements that should be met should help in deciding which planner to go with.

2) Planning with Heuristics

2.1)

a)

Planning with Heuristics

2.1) a) h_1 and h_2 are 2 consistent heuristic functions

$\Rightarrow h_1(s_{goal}, s_{goal}) = 0$

& $h_1(s) \leq c(s, succ(s)) + h_1(succ(s)) \quad \forall s \neq s_{goal}$

Similarly,

$h_2(s_{goal}, s_{goal}) = 0$

& $h_2(s) \leq c(s, succ(s)) + h_2(succ(s))$

Now, for a heuristic, $h(s) = \min(h_1(s), h_2(s))$

$h_*(s_{goal}, s_{goal}) = \min(h_1(s_{goal}, s_{goal}), h_2(s_{goal}, s_{goal}))$

$= \min(0, 0) = 0$

&

$h(s) = \min(h_1(s), h_2(s))$

$\leq \min(c(s, succ(s)) + h_1 succ(s), c(s, succ(s)) + h_2 succ(s))$

$\leq c(s, succ(s)) + \min(h_1 succ(s), h_2 succ(s))$

$\Rightarrow \boxed{h(s) \leq c(s, succ(s)) + h(succ(s))}$

Hence proved $h(s)$ is consistent

b)

b)

$h(s_{goal}) = \max(h_1 s(goal), h_2(s_{goal}))$

$= 0$

i.e. $h(s) = \max(h_1(s), h_2(s))$

$\leq c(s, succ(s)) + \max(h_1(succ(s)), h_2(succ(s)))$

{ Similar to previous proof.

$\Rightarrow \boxed{h(s) \leq c(s, succ(s)) + h(succ(s))}$

$\therefore$ $h(s)$ is consistent

Hence Proved

2.2) b

2.3) f

2.4) e