1. Conceptual Questions

(a) Explain the difference between simple and true proportional navigation. When would 5% simple PN be used over TPN?

A simple proportional navigation also known as a SPN is based on maintaining a constant angle between the line of sight to target and the velocity vector. This is a much simpler design and implementation of a TPN, a TPN is an enhanced version of a PN, the major difference is that a TPN considers the changing maneuverability that the target will have. A 5% simple PN would be used when target may have some degree of maneuverability but not as intense to implement a TPN.

(b) If you have a single starting point and 20 stops on your delivery run (traveling delivery 5% man). How many possible paths are there that visit each of the stops? Make sure to show your work.

To complete this question we use this equation $P(n) = n!$ Since we have 20 we take $P(20) = 20! = 2432902008176640000.$

(c) Describe how the genetic algorithm operates, and how it is applicable to the traveling 5% salesman problem.

The genetic algorithm is an algorithm that looks for best suitors or targets, the way the algorithm starts is by first having a generated map and destinations, each node on the map will then be given a value of some sort depending on what the search is for. These nodes are then placed into populations, these populations have chromosomes, and they then have genes inside them. The genes can be considered nodes, a chromosome can be seen as a path or location and population can be the destination.

When a node is scored it is put into a chromosome, these chromosomes will then be considered a parent node, a parent node can mate with another parent and these parents create offsprings, these offsprings can be represented as a better solution then the parent. The GA algorithm will then repeat the mating and off springs until it reaches the best solution it can compute.

For a TSP problem the GA algorithm can label each destination as a population and we can label each destination as a gene. Depending on the route taken the genes can be labeled as tour cost or whatever cost wanted, we then place the order of locations as a chromosome. We would want lower cost so the chromosomes will be found in order of least to most expensive. As the search grows more routes or chromosomes will appear and may erase previous chromosomes found. Once finished we will have the best costing route.

(d) Describe 3 advantages and 3 disadvantages in using ROS (compared to custom robotics 5% frameworks).

Advantage:

- ROS today is the most used and common robot operating system today, so many GitHub and other sources are built on its system allowing for easier access for everybody.
- ROS system is easily accessible and has various libraries and packages prebuilt to allow user learn new skills faster.

- ROS system is built an easy to maneuver publish and subscriber model, this allows users or other coders to work across various software such as Python or C, making it easier to distribute the system or design.

Disadvantages

- ROS has a difficult learning curve and can at times be computationally heavy, for beginners like us in this class it was very frustrating and difficult to sometimes figure out problems.
- ROS can also have too much communication, using the subscriber and publish model if a system has too much data it can affect the performance of the operating system. Causing overhead.
- It is difficult to customize or create new systems without an deep understanding of how ROS works, this limits the users creative ability if he doesn't understand the system as much.

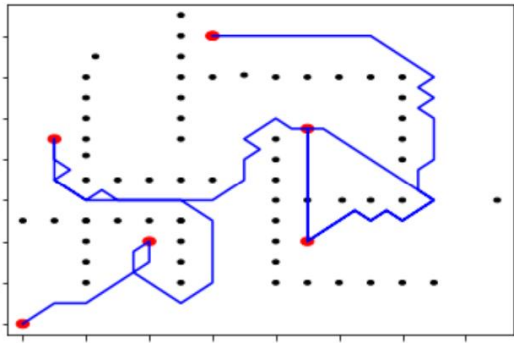
2. Trajectory Generating and Following Evader

Your objective is to incorporate your True Proportional Navigation pursuer turtlebot with an evader that generates and follows an optimal path from a start to the goal. You must provide a plot of each bots' path along with the desired trajectory generated for the evader on a single plot. You are not required to include any obstacle avoidance into your pursuer path planning, but may find some small modifications are required to successfully chase (and catch) the evader prior to the evader reaching the goal location as well as make sure your evader does not collide with any obstacles. In addition to the plot(s), provide the code used for the problem, and provide any comments/discussions to interpret your results and describe any modifications you had to make in order to be successful.

3. Traveling Salesman Problem with A*

Your objective is to use the A* Algorithm to navigate through the modified maze (obstacle list posted below) to deliver five packages. You must start at (0,0) and visit each of the delivery points. Provide a plot showing the obstacles, A* paths, and delivery locations. Provide the total travel distance of your solution. How does this compare with the other possible paths (how good is the solution, are there other paths that have much higher travel costs)? Turn in a copy of your working code.

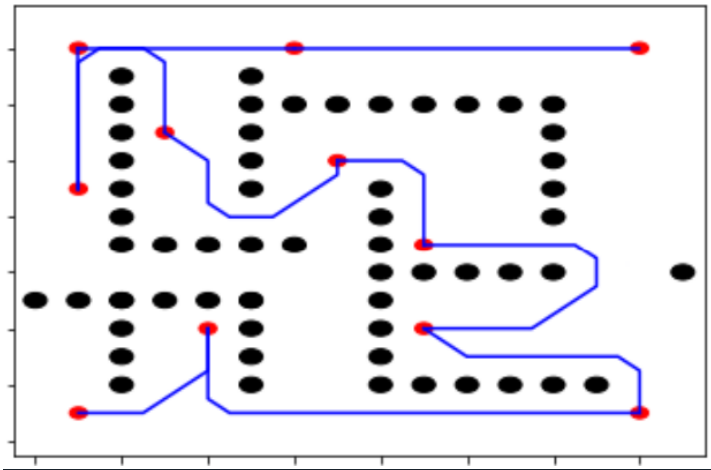
```
In [47]: runfile('C:/Users/mario/OneDrive/Desktop/
Luis_Rivas_unmanned_systems_2023/Working Codes/Problem3E2.py', wdir='C:/Users/
mario/OneDrive/Desktop/Luis_Rivas_unmanned_systems_2023/Working Codes')
Reloaded modules: Dijkstras, AStarAlgorithm
path taken (0, 0) (9, 4) 200 0 200
path taken (0, 0) (4, 4) 42 0 43
path taken (0, 0) (1, 9) 218 0 218
path taken (0, 0) (9, 9) 218 1 219
path taken (0, 0) (6, 14) 528 0 529
path taken (9, 4) (4, 4) 223 0 224
path taken (9, 4) (1, 9) 312 0 312
path taken (9, 4) (9, 9) 73 0 73
path taken (9, 4) (6, 14) 270 0 271
path taken (4, 4) (1, 9) 174 0 174
path taken (4, 4) (9, 9) 154 1 155
path taken (4, 4) (6, 14) 456 0 456
path taken (1, 9) (9, 9) 141 1 142
path taken (1, 9) (6, 14) 363 0 363
path taken (9, 9) (6, 14) 202 0 202
```



top figure shows the path taken and the value to travel to each location, second image shows the plot.

4. Super TSP - Brute Force

You have a map very similar to Problem 2 (just a few obstacles removed), with many more waypoints that must be visited. The vehicle must start from the first delivery point (1,1). Use your A*-based TSP code to identify the best path. Plot identical to Problem 2. Please note this will take 10 minutes up to 60 minutes to compute (based on your computer performance). It is recommended to use the tqdm package to keep track of timing. Simply put "from tqdm import tqdm" at the top, and in your big for loop put "for i in tqdm(range(0,len(perm list))):"



```
In [49]: runfile('C:/Users/mario/OneDrive/Desktop/
Luis_Rivas_unmanned_systems_2023/Working Codes/Problem4E2.py', wdir='C:/Users
mario/OneDrive/Desktop/Luis_Rivas_unmanned_systems_2023/Working Codes')
Reloaded modules: Dijkstras, AStarAlgorithm
Number of times computed:  55
Number of paths found:  39916800
39916800it [03:49, 173611.49it/s]
Time taken 229.93878650665283
best path ((1, 1), (4, 4), (14, 1), (9, 4), (9, 7), (7, 10), (3, 11), (1, 9),
(1, 14), (6, 14), (14, 14))
min cost 611.8399999999999
```

Top picture shows path taken bottom has time computed, paths, time taken and cost, I tried to get a rounded answer for both time and cost but was met with errors.

GitHub Link to Codes