

ENPM661 - Planning for Autonomous Robots

Final Exam Spring 2024

Name: Suhas Nagaraj

UID: 119505373

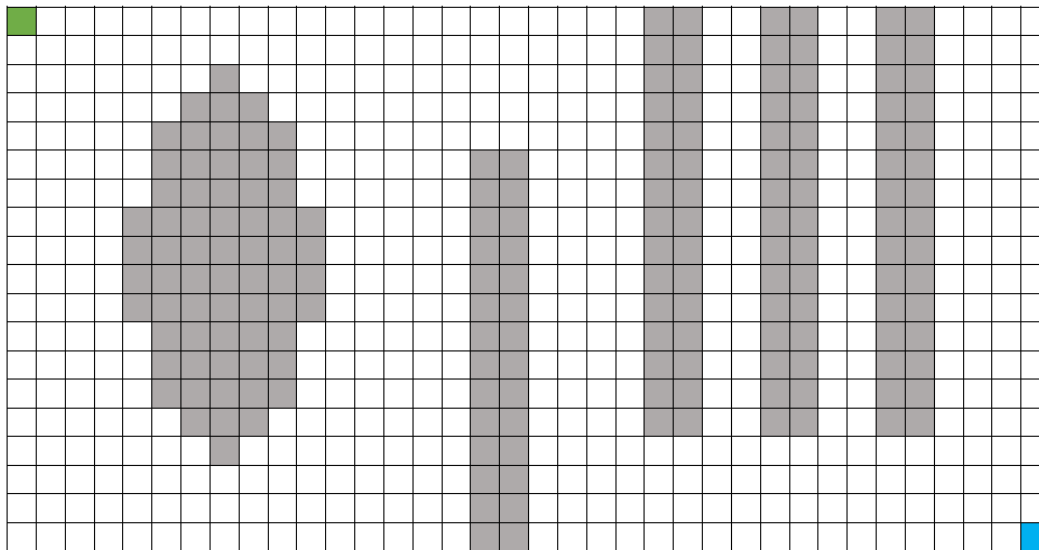
Directory ID: suhas99

Question 1 (6 marks)

Employ the use of Bug Algorithm and Bug2 algorithm to go from start point (green) to end point (blue). Present your answers as a diagram. Explain any differences you see between the two. To learn about the two algorithms, read the following:

http://www.cs.cmu.edu/~motionplanning/lecture/Chap2-Bug-Alg_howie.pdf

Remember that the bug agent can only sense its immediate surroundings, which is an 8-cell region in our case (2 top and bottom, 2 left and right and 4 diagonals).

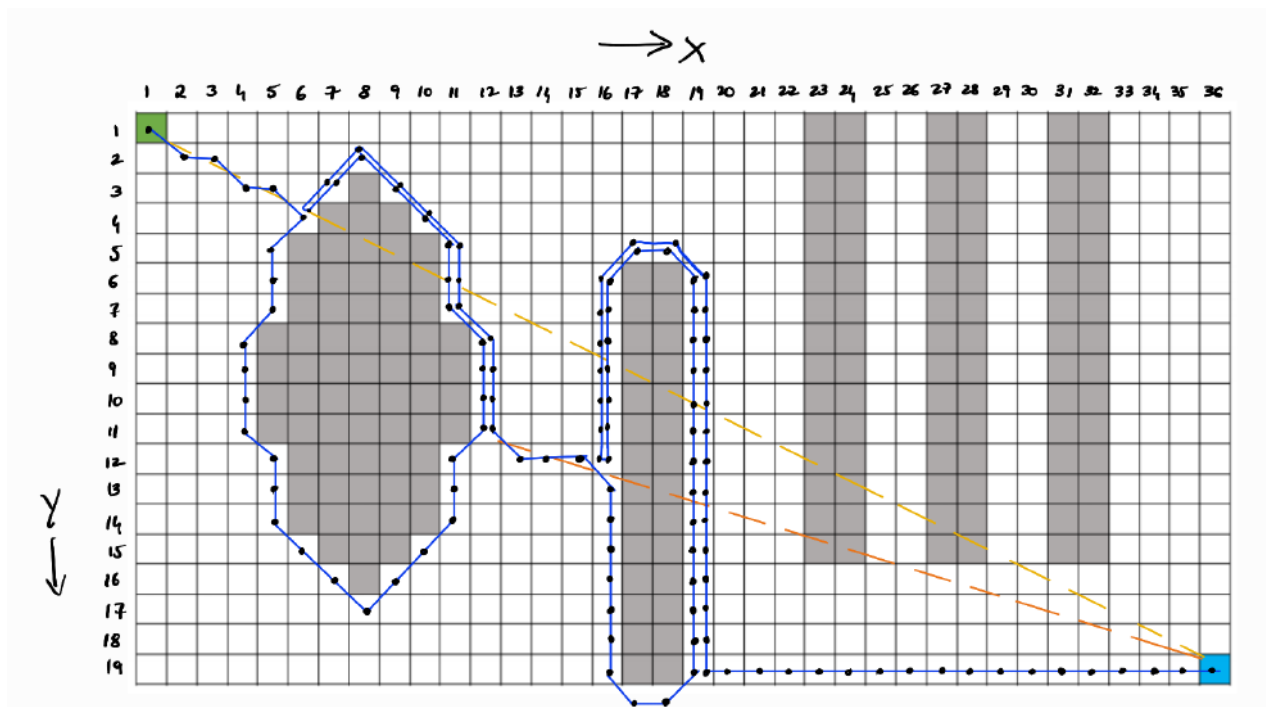


Answer:

Assumptions made:

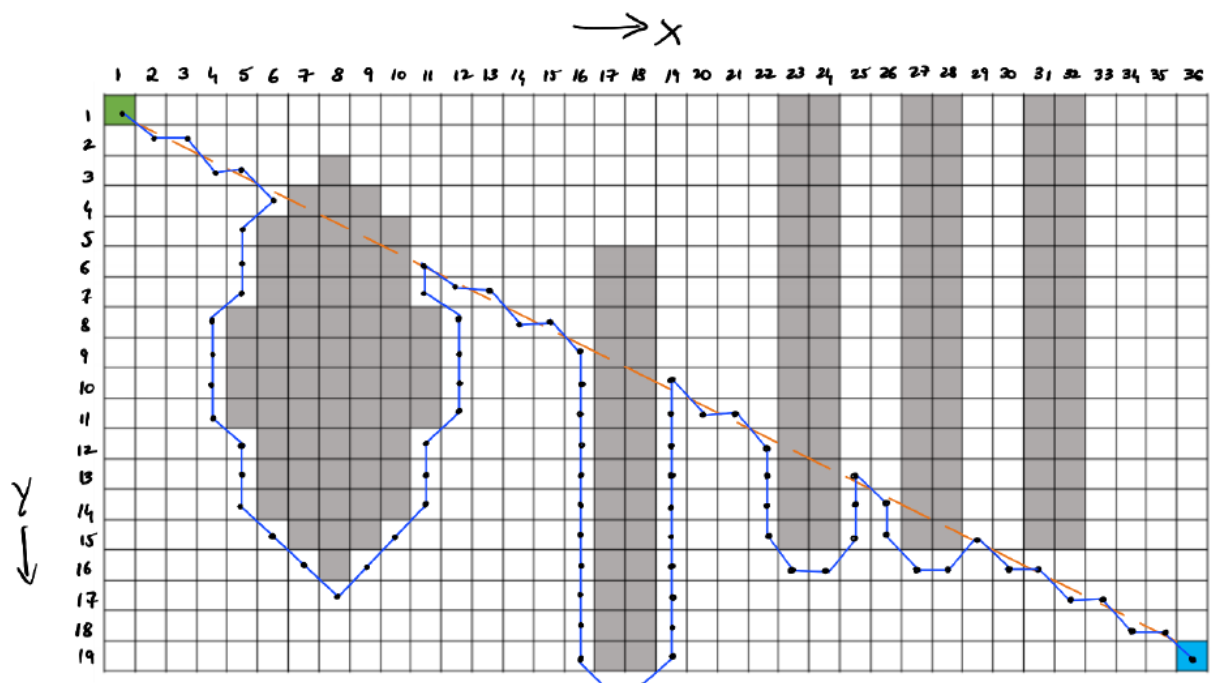
1. It is assumed that the map is not bounded by outer walls and the bug agent can move beyond the grid space presented in the diagram (above and below the obstacles).
2. It is assumed that the bug agent can show 8 possible motions: up, down, right, left, and the four diagonals.
3. It is assumed that the robot makes a right turn when it meets an obstacle.

Bug 1 Algorithm diagram:



In the above diagram, the dotted yellow and orange lines are drawn for reference, they do not have any significance. The distance to goal from each point in the robot's circumnavigation path was computed to find the robot's exit point. For example, the cell (12,11) was the closest to the goal and hence the robot continued its journey to the goal from this point.

Bug 2 Algorithm diagram:



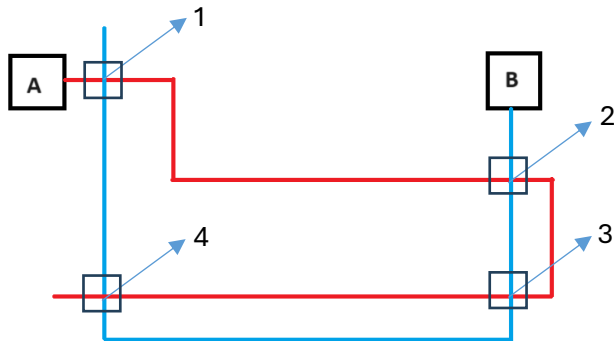
In the above diagram, the dotted line drawn in orange is the m-line that the robot is following.

Observations made:

- In the given map, bug 2 algorithm is observed to be more efficient than bug 1 algorithm in terms of total distance travelled to reach the goal.
- In bug 1 algorithm, the bug agent moves around the obstacle (circumnavigate) to find the closest free point to the goal. It then returns to this point to continue its path towards the goal. This process is repeated at every obstacle it encounters until it reaches the goal.
- In bug 2 algorithm, the bug agent moves along a 'm-line'. The m-line is the line drawn from the initial point to the final point and the agent tends to keep this line as a reference and follow this line to reach the goal. When the agent encounters an obstacle, it circumnavigates the obstacle until it finds itself at a point on the m-line and closer to the goal than the point it left. Then it resumes following the m-line and repeats this process until it reaches the goal.
- The bug agent is observed to be retracing the path to return to the closest point to the goal in bug 1 algorithm. This is not observed in bug 2 algorithm.
- In bug 1 algorithm, the agent moves directly to the goal after it exits the second obstacle. But in bug 2 algorithm, the agent encounters 3 more obstacles before reaching the goal.

Question 2 (6 marks)

Sketch the state space for the two robots moving along the fixed paths shown below and show a velocity tuning graph. Then show a feasible solution to avoid contact between these two robots.



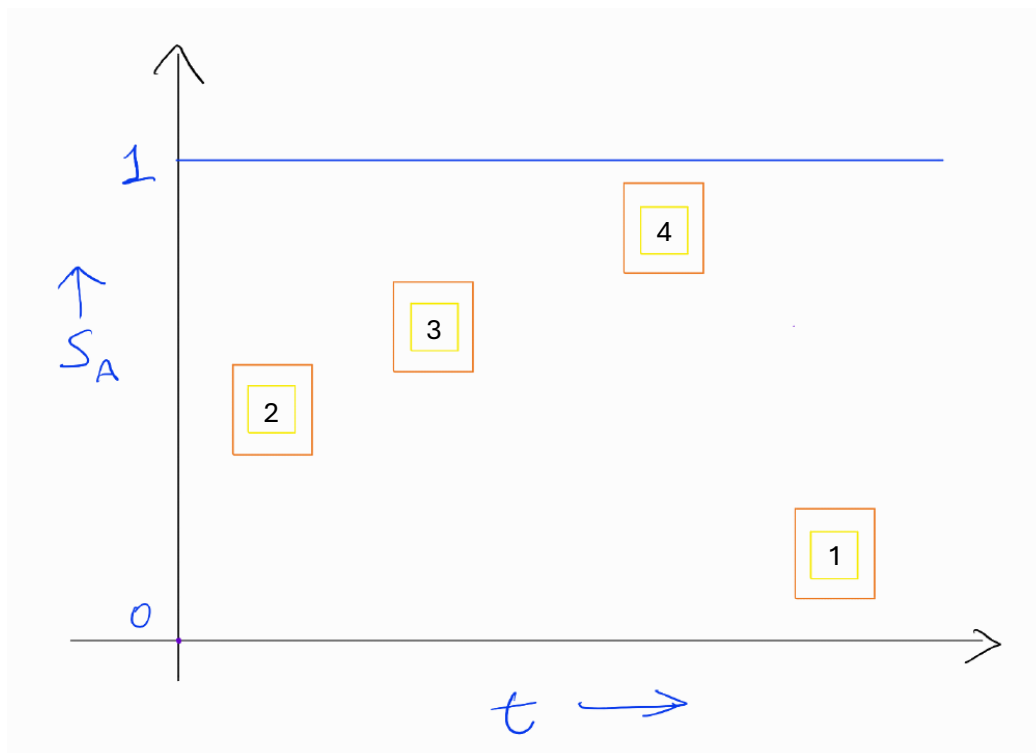
Answer: The distance and time values are visually approximated to draw the state space diagram.

Part 1: Velocity of Robot B is constant, and velocity of Robot A is tuned.

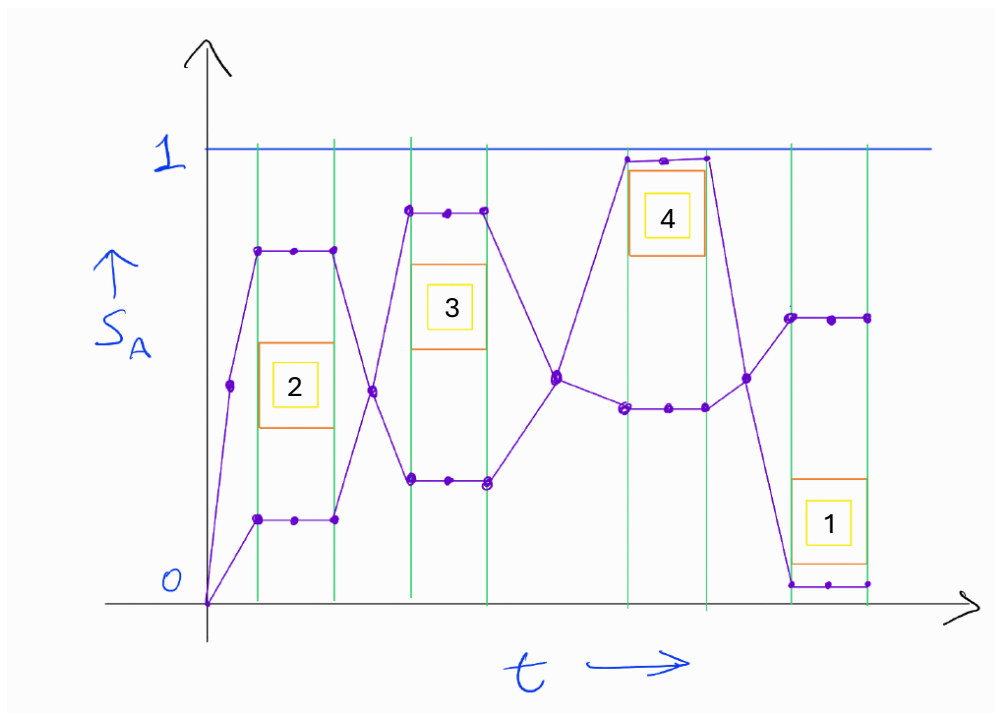
Assumptions:

- It is assumed that the Robot B moves with constant velocity and Robot A's velocity is tuned accordingly to avoid collision between the 2 robots.
- This problem is considered as a robot-dynamic obstacle problem where the Robot B acts as a dynamic obstacle for Robot A and hence the velocity of Robot A is tuned accordingly.

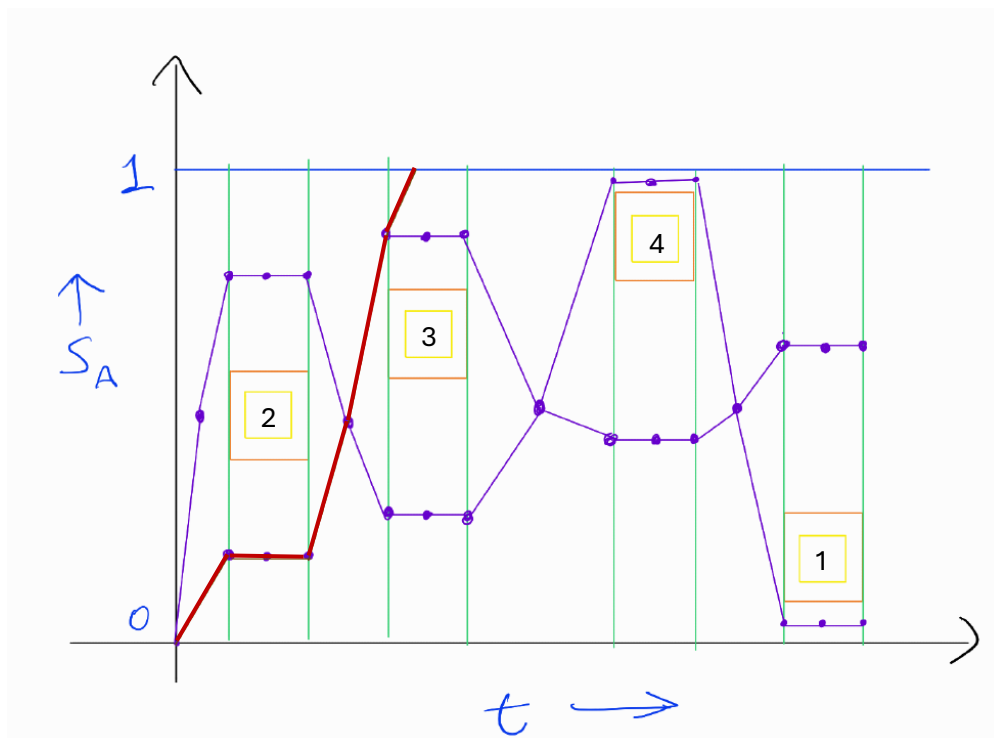
State Space Diagram:


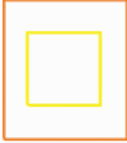




Velocity tuning graph: By vertical cell decomposition method



Feasible solution:



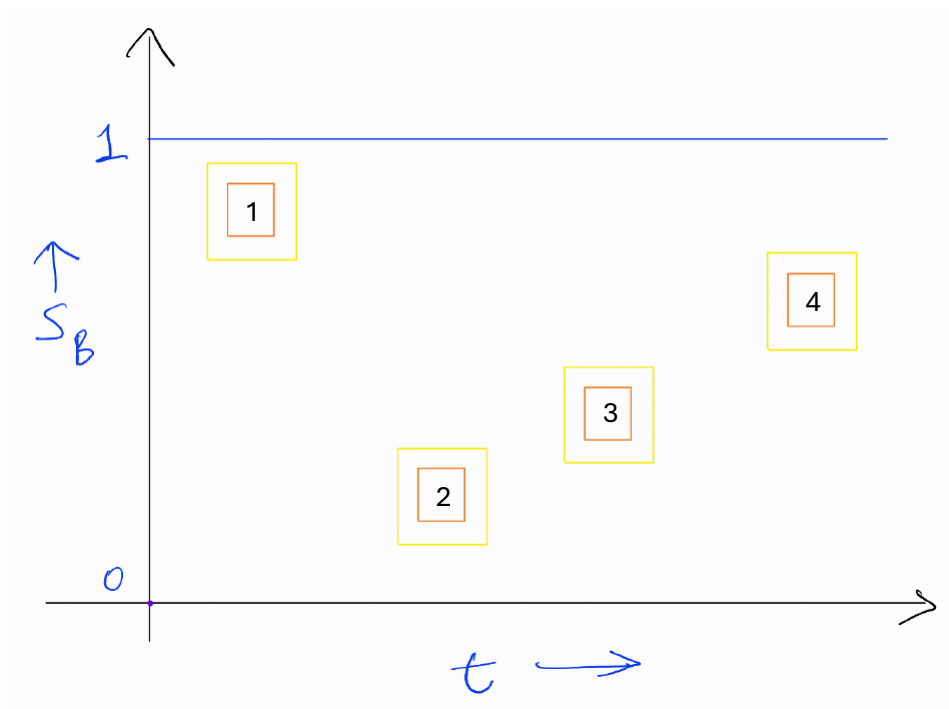
Feasible solution	
Intersection / Collision points in s-t domain	
Connections obtained from vertical cell decomposition	
Line representing vertical cell decomposition	

Part 2: Velocity of Robot A is constant, and velocity of Robot B is tuned.

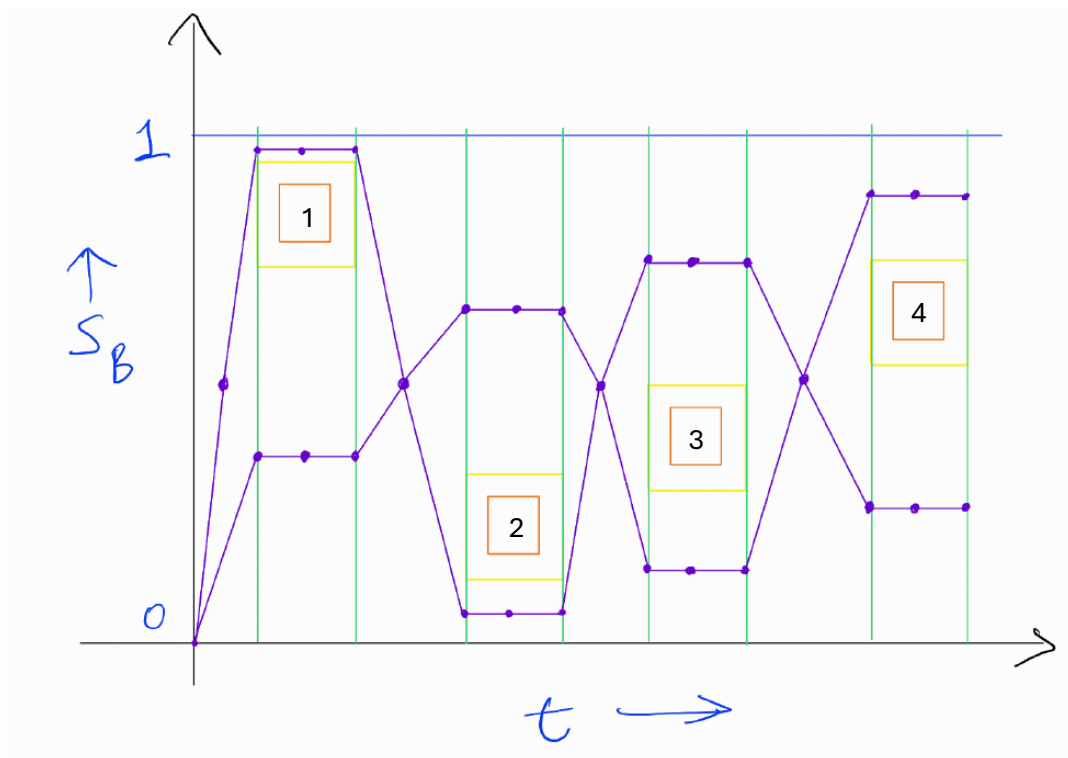
Assumptions:

- It is assumed that the Robot A moves with constant velocity and Robot B's velocity is tuned accordingly to avoid collision between the 2 robots.
- This problem is considered as a robot-dynamic obstacle problem where the Robot A acts as a dynamic obstacle for Robot B and hence the velocity of Robot B is tuned accordingly.

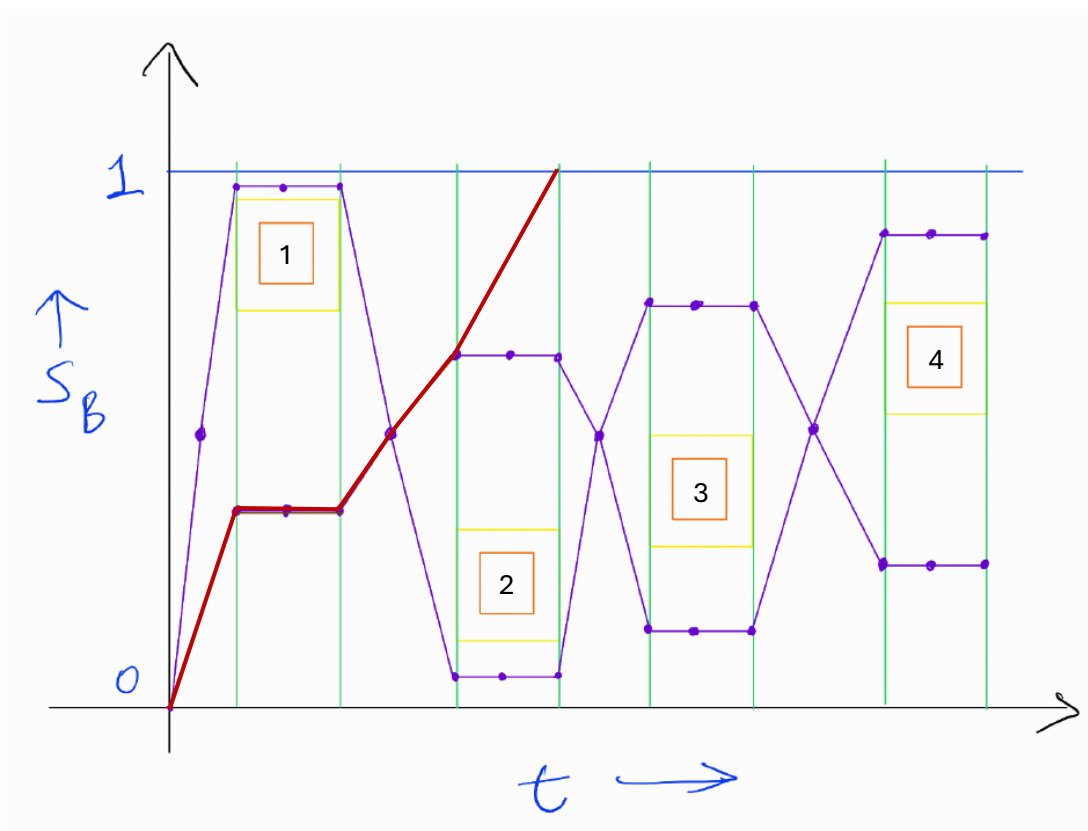
State Space Diagram:


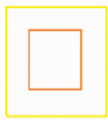




Velocity tuning graph:



Feasible solution:

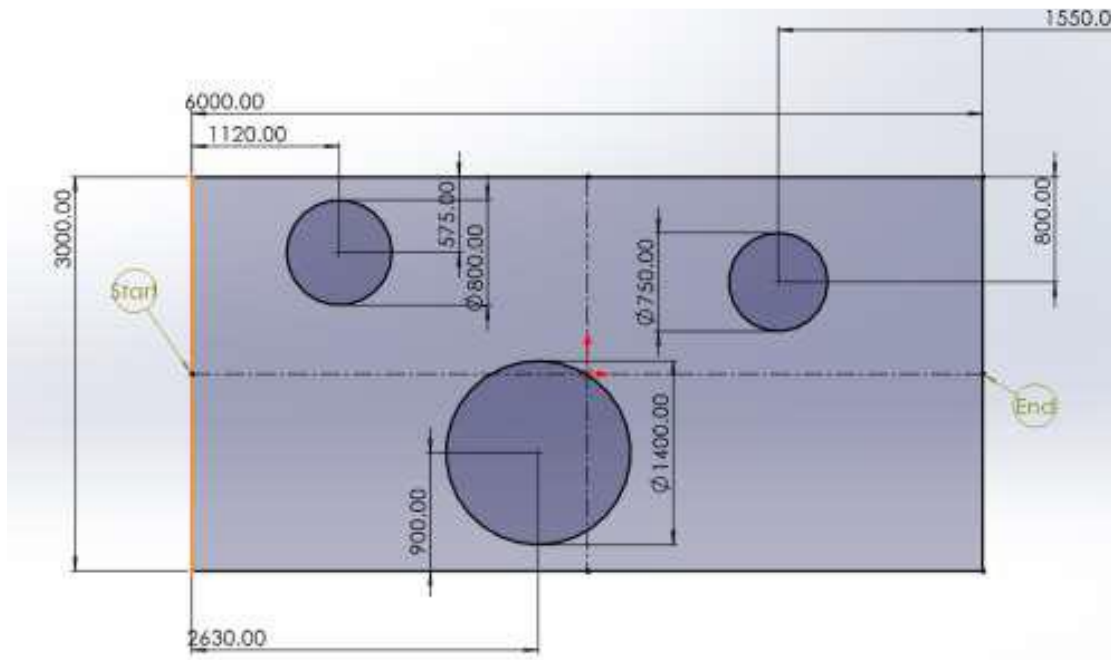


Feasible solution	
Intersection / Collision points in s-t domain	
Connections obtained from vertical cell decomposition	
Line representing vertical cell decomposition	

- The state space diagram is the obstacle position vs time graph drawn for an agent.
- In part 1, I have considered robot A to be the agent and robot B to be the obstacle. The robot B moves in constant velocity whereas the robot A's velocity is tuned/selected/decided to avoid collision.
- In part 2, I have considered robot B to be the agent and robot A to be the obstacle. The robot A moves in constant velocity whereas the robot B's velocity is tuned/selected/decided to avoid collision.
- Vertical cell decomposition is carried out to generate a roadmap for velocity tuning. The slope of the lines represents the velocity. By considering various factors like the max velocity of the robot, time taken to reach the goal, etc, the velocity profile of the robot which is being tuned is selected.

Question 3 (10 marks)

Find a solution for the following path planning problem using bi-directional A* algorithm and plot your results.



Now, solve the same problem using a potential force-field implementation. In short, the robot and the obstacles are assumed to be charged positively while the goal is negatively charged. This causes the robot to move towards the goal while avoiding the obstacles.

https://www.cs.cmu.edu/~motionplanning/lecture/Chap4-Potential-Field_howie.pdf

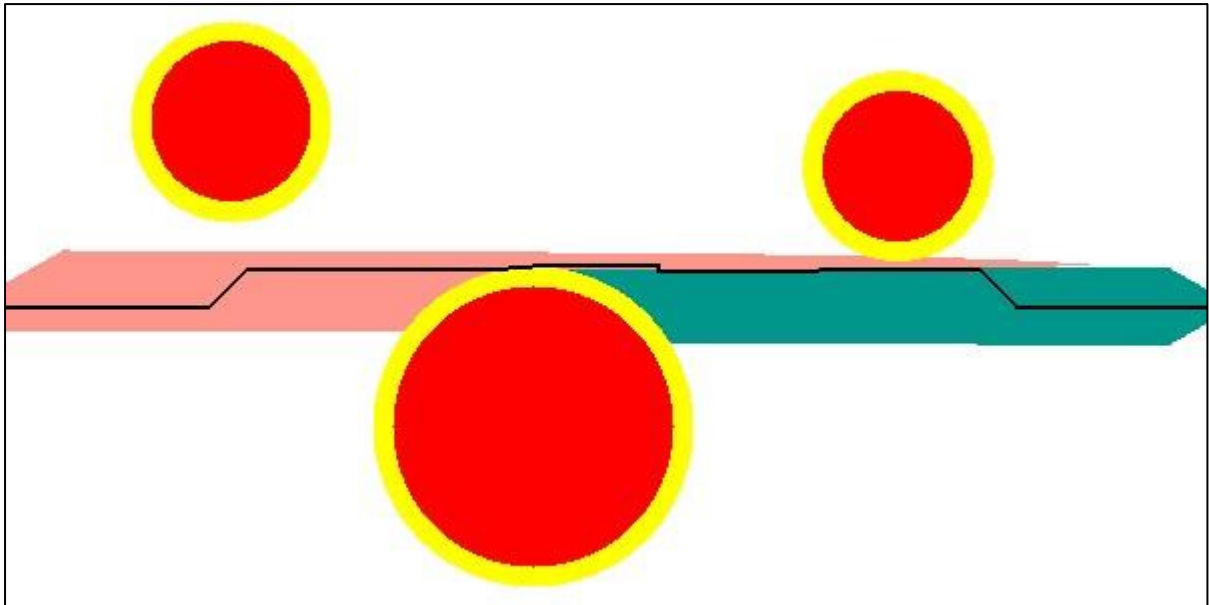
Your implementation can be a stepwise process where the robot moves a fixed length in the direction of the resultant instantaneous force, and the forces are recalculated with every iteration. The following variables will have to be selected and tuned by you:

- Charge on the robot,
- Charge on the goal,
- Charge density on the obstacles,
- Anything else that you might need

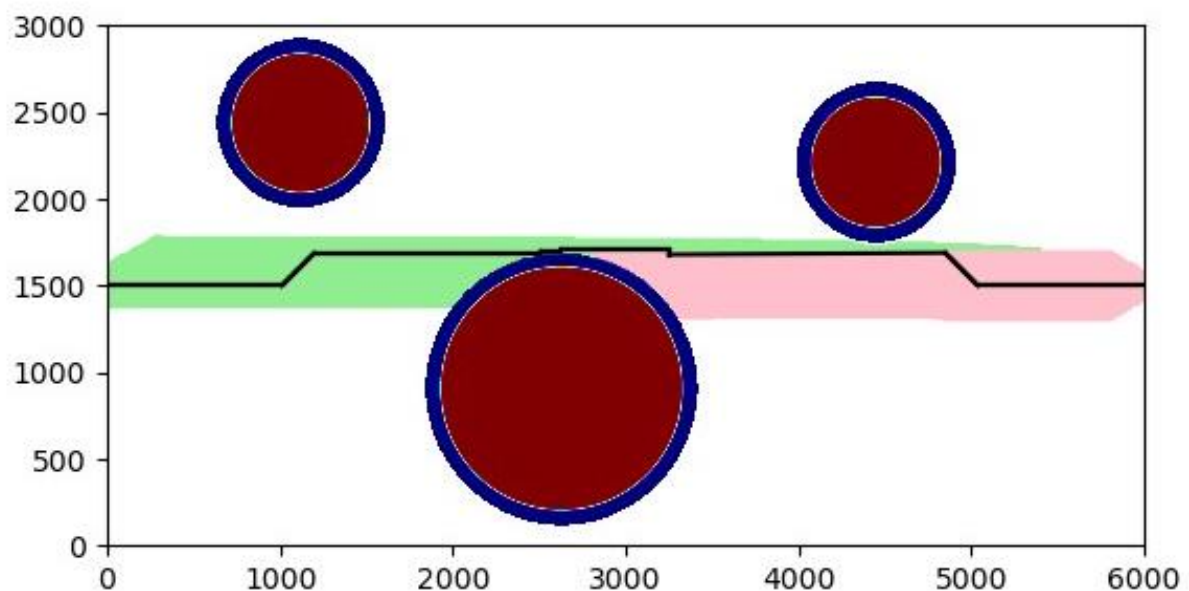
Compare your plot with the bidirectional A* results.

Answer:

Bidirectional A* path plot (pygame):



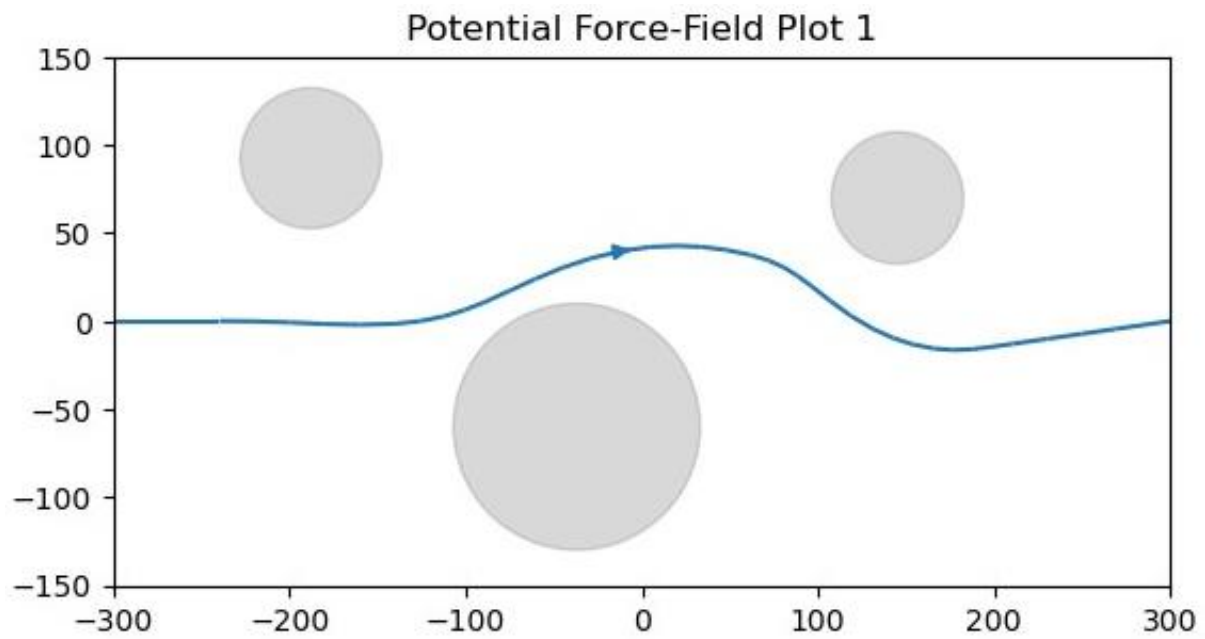
Bidirectional A* path plot (Matplotlib):



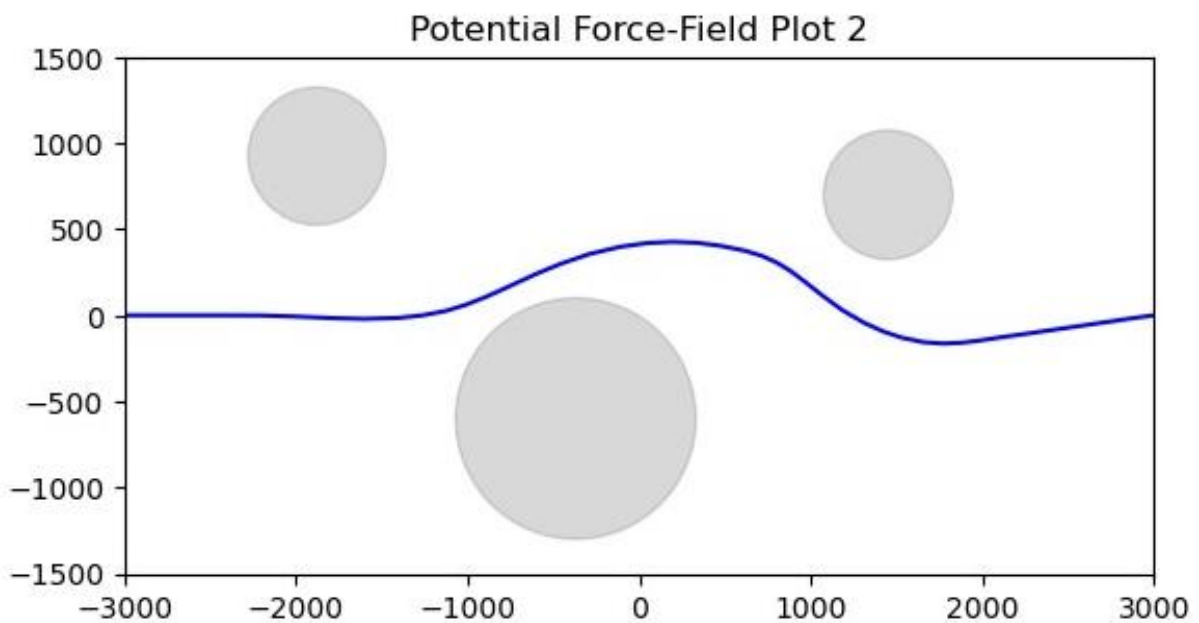
Parameters considered:

1. Weight = 1
2. Action set = (right, left, up, down, up-right, up-left, down-right, down-left)
3. Cost to goal = Euclidean distance
4. Clearance/offset = 100 mm
5. The map origin is at (3000,1500) with respect to plot origin.

Potential Force-Field path plot (Vector flow streamline):



Potential Force-Field path plot (Path obtained):



Parameters considered:

1. Distance at which the repulsion from obstacle is experienced, $\text{force_range} = 600 \text{ mm}$
2. Goal threshold = 10 mm
3. Attraction coefficient = 50
4. Repulsion coefficient = -200

Observations and comparison:

1. The path obtained from Bi-Directional A* seems to be more optimal than the path obtained from potential field algorithm.
2. The Bi-Directional A* algorithm has higher space and time complexity as it involves storing nodes and node related values, computing costs, and searching through lists or sets. Whereas the potential field algorithm is more computationally efficient, especially in simple maps with less obstacles, as it stores only force values and iterates through the cells only once.
3. Bi-Directional A* algorithm employs action sets to generate new nodes while potential field algorithm does not use any action set. Hence, Bi-Directional A* is more suited for discrete state spaces whereas potential field algorithm is suited for continuous state spaces.
4. There is a tendency of the agent to get stuck in local minima and never reach the goal in potential field algorithms. Whereas A* algorithm is almost complete with right choice of heuristic function.
5. In the path plots that I obtained, the path generated by Bi-Directional A* is straighter and is more optimal in terms of cost to reach the goal. But the path generated by field algorithm is smoother and more continuous and there is no sudden change in direction.
6. The Bi-Directional A* algorithm takes 9-10 seconds to generate a solution whereas the potential field algorithm gives a solution in 3-5 seconds.
7. In conclusion, Bi-Directional A* gives more efficient path but potential field algorithm is computationally efficient.

Note: The codes are submitted along with the pdf. The suhas99_BiAstar.py file contains the code for Bi-Directional A* algorithm execution whereas suhas99_PotentialField.py file contains code for potential field algorithm execution. Please make sure numpy, heapq, matplotlib, math, pygame libraries are installed before running the code.

Question 4 (8 marks)

Summarize the following paper in one page, focusing on the motivation, method, advantages, results, and conclusion. (The paper will be linked in assignment description / in announcements)

<https://ieeexplore.ieee.org/document/9636687>

Answer: Summary of the paper:

The paper titled “MR-iSAM2: Incremental Smoothing and Mapping with Multi-Root Bayes Tree for Multi-Robot SLAM” goes over a new data structure, multi-root Bayes tree (MRBT), and a new incremental optimization algorithm, multi-robot iSAM2 (MR-iSAM2), proposed for SLAM inference problems involving multiple robots.

Motivation: The main motivation behind developing these new methods is that in multi-robot cases, updating the new measurements taken by different robots in the single root of a regular Bayes tree data structure destroys the sparsity and makes the iSAM2 algorithm less efficient. As Bayes tree is directed, the new measurements taken by different robots must be incorporated at the top of the tree to ensure efficient updates. Due to this, during the optimization step, the recent state of all the robots will be eliminated last resulting in large cliques containing states from different robots, making the algorithm inefficient. In contrast, if the measurements from only one robot is incorporated at the root, updating the tree with respect to the measurements made by other robots can be expensive as these other new measurements will be incorporated far away from the root. Hence, to overcome the limitations and drawbacks of the iSAM2 and Bayes tree data structure with respect to multi robot SLAM problems, MRBT and MR-iSAM2 are proposed.

Method: MRBT is an extension of Bayes tree data structure, where instead of a tree with a single root, MRBT has multiple Bayes trees with individual roots, packed into one, and with the same undirected clique structure. The MR-iSAM2 algorithm inherits the incremental update feature of iSAM2 and uses MRBT to represent the belief state and maintains the MRBT in each incremental update. A root is created for each robot and this root contains the recently accessed variables by that robot. An incremental update is performed for each robot along the corresponding Bayes tree starting at its root. When new measurements taken from a robot is added to the corresponding root in MRBT, the root is recomputed, and this new information added changes the belief state of the variables as it influences the clique structure of the MRBT. Generally, these new measurements have local effects as only the recently accessed variables are part of the measurement. Hence, a small fraction of the cliques around the root is recomputed. But there can be cases where these local effects can have a global influence as the measurement by one robot can influence the estimation done by other robots. Hence, the effect of the new measurement made by one robot is propagated to the other roots of the tree. This is achieved through dual-directional edges on the path among the roots.

Advantages: As in MR-iSAM2 with MRBT each robot has its own root, the information processing and incremental update is more efficient and less expensive. iSAM2 algorithm stacks the measurements made by all the robots at a same single root resulting in large cliques. Whereas in MR-iSAM2, the measurements made by different robots are added at their corresponding roots, resulting in smaller cliques by growing the bayes tree in different branches for different robots. For example, if multiple robots are taking measurements from different regions of the map, this information can be incorporated at different roots and the tree is grown separately. This reduces the complexity of the tree and results in more efficient data storage and computation. This can be further explained by giving an example of a building being built. The electrician, plumber, mason etc all focus on different aspects

of the construction which results in a more efficient process than a single person taking care of all aspects of the construction, which makes the process more complex.

Results: The authors compare the efficiency of the MR-iSAM2 algorithm against the regular iSAM2 algorithm using 2 different datasets. The information/measurements by different robots are incorporated at a single root in iSAM2 whereas the same is incorporated in different specific roots in MR-iSAM2. The first dataset used for comparison is CityTrees dataset which has a single robot's odometry and landmark observations. The dataset was modified by dividing it into different parts representing measurements by multiple robots and the comparison was done for different robot number scenarios. MR-iSAM2 was observed to be ten times more efficient than the iSAM2 for 2 robot scenario and the ratio increases even further as the number of robots increases. This is mainly because the number of recalculated cliques and re-linearized variables at each update is smaller in MR-iSAM2 than in iSAM2. It was also observed that MR-iSAM2 resulted in smaller clique size as information is processed at different roots. The accuracy of the MR-iSAM2 algorithm was same as the iSAM2 algorithm indicating proper updating and data handling. The second dataset used for comparison is the UTIAS Dataset which contains the odometry and range measurements taken by 5 robots in lab environment at 15 different landmarks. Results observed with the UTIAS Dataset was like the results observed with the CityTrees dataset. The MR-iSAM2 was seen to be at least 20 times faster than iSAM2 with the same accuracy. The authors also pointed out that the maximum number of cliques observed the 2 algorithms were similar in both the cases and they justify this by explain the reason that the number of cliques depend on number of features/landmarks in the dataset. As the robots observe the same landmarks at different times, the number of cliques stays the same. But they expect to see a difference in the number of cliques when the robots explore different parts of the map and mutual observation is less frequent.

Conclusion: In conclusion, the MR-iSAM2 algorithm presented in the paper along with a new data structure MRBT is tested and proved to be more efficient than the regular iSAM2 algorithm for the SLAM problems involving multiple robots. The multi-root feature of MR-iSAM2 algorithm is observed to be better at handling measurements taken by the multiple robots as it reduces the clique size and complexity of updating the tree. The paper clearly explains the modifications done on the Bayes tree data structure to get the MRBT and the modifications done on iSAM2 to get MR-iSAM2 and the experiments performed to compare the two algorithms. This paper further opens future work opportunities to perform the incremental updating step parallelly to further improve the efficiency.