SWARM ROBOTS FOR MAPPING AND PATH PLANNING

JACK GLADOWSKY, ISHAAN DESAI, WILLIAM FOX

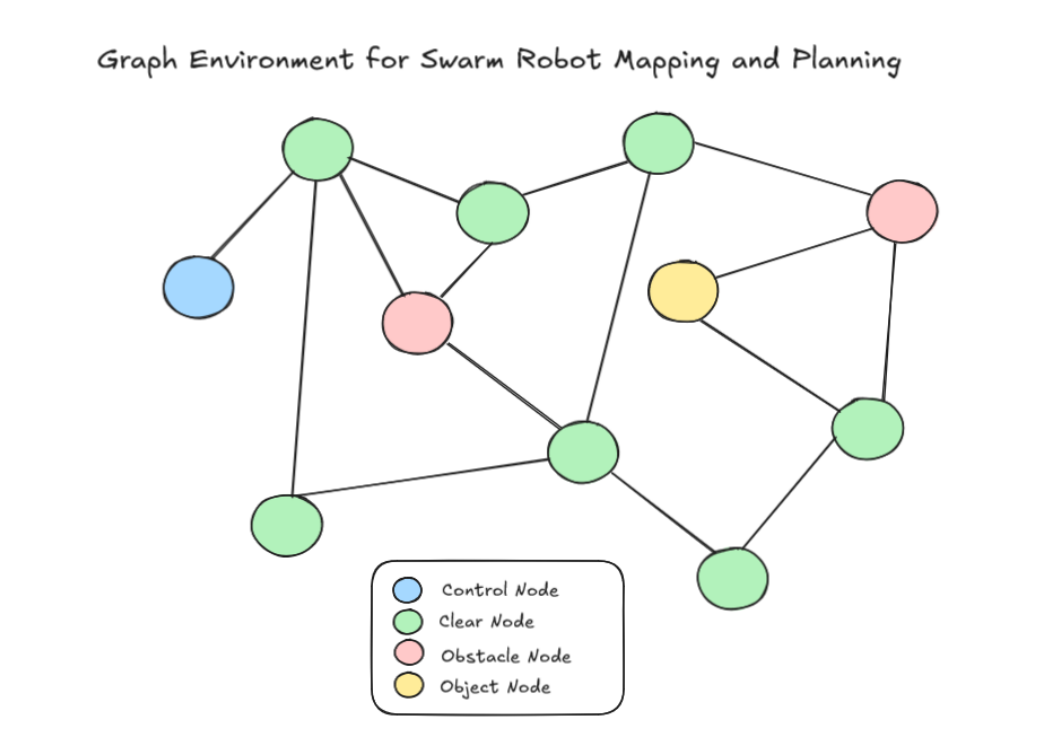
PROJECT SCOPE AND OBJECTIVES

Project Scope

- Develop a system for generating graph environments for robots to traverse and map, so that they can retrieve various objects throughout the environment.

OBJECTIVES

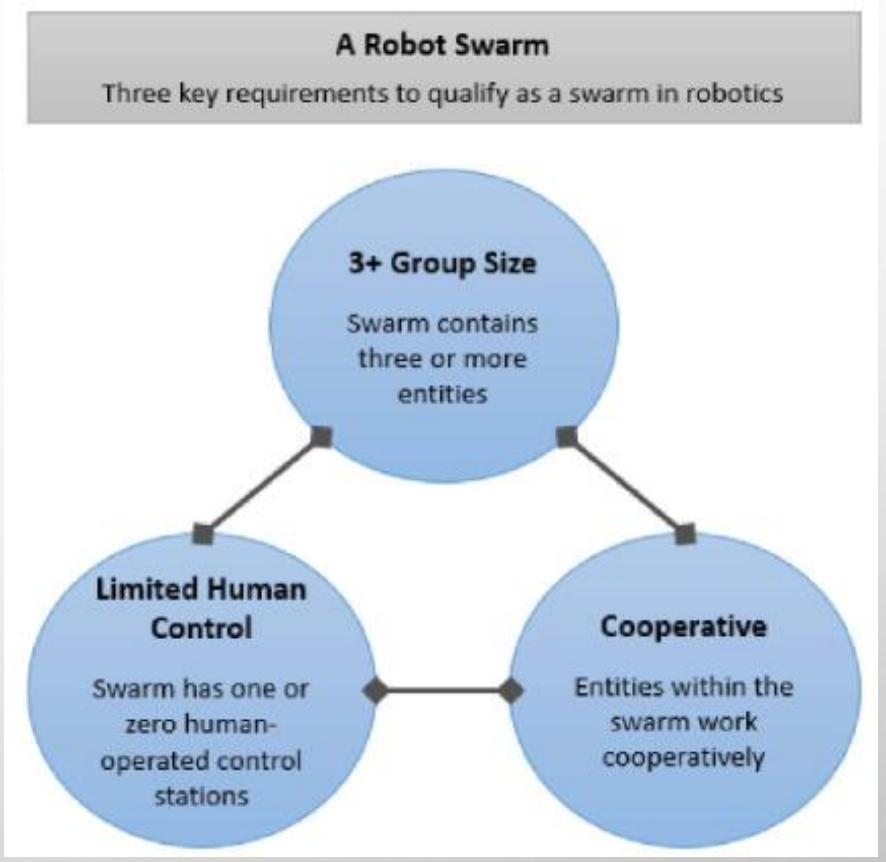
1. Develop efficient mapping algorithm for mapping graphs
2. Implementing optimized path planning strategies
3. Generating graphical representation of an explorable environment



LITERATURE REVIEW

- What is a Robot Swarm: A Definition for Swarming Robotics – LINK
  - Defines a robot swarm as a group of 3+ entities, the entities work together, and the swarm has limited human control
  - 'Many real-world problems can be addressed through use of swarms'

- Real Applications for Swarm Robotics Applications – LINK
  - Swarm robotics can coordinate multiple robots for search-and-rescue missions, navigating complex environments to locate survivors and deliver assistance efficiently, not path constrained.
  - Groups of robotic units can work together to monitor crop health, plant seeds, and optimize irrigation, enhancing productivity while minimizing resource use.



METHODOLOGY – NODES AND GRAPHS

- A graph is just a list of nodes, where each node keeps track of its neighbors
- Nodes are generated with states assigned randomly (80% chance of Clear, 10% Obstacle, 10% Object) with 1 control point
- Nodes also store their respective neighbors, allows for robot to know where to go next
- Using stacks to hold node neighbor

Graph Generation Pseudocode

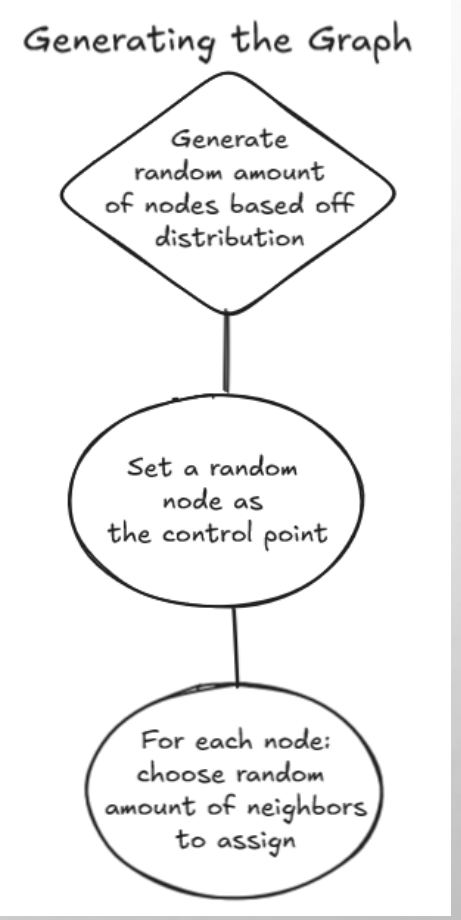
1. Loop through the number of nodes and generate randomly based on state distributions
2. Set one nodes state as the control point
3. Loop through all the nodes again
    A. Choose a random number of neighbors
    B. Loop through the number of neighbors and choose a random node to add as a neighbor
    C. Check if the neighbor is already a current neighbor
    D. Push each node to the neighbor list of each other

Time Complexity: O(n*m)

enum NodeState:
- CLEAR = 0
- OBSTACLE = 1
- OBJECT = 2
- CONTROL = 3

class GraphNode:
- int NodeID
- NodeState state
- Vector<GraphNode*> neighbors

class Graph:
- int numNodes
- Vector<GraphNode*> nodes
- GraphNode* controlPoint



METHODOLOGY – ROBOTS AND CONTROLLER

Controller::Map Function

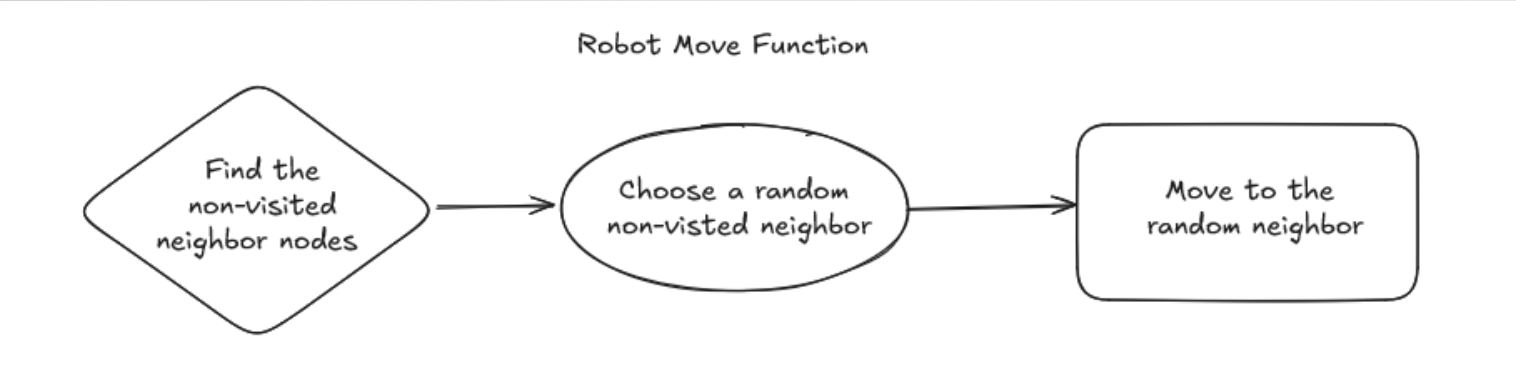
- Create the robots
- Move robots to control point
- Initialize *exploredMap* with control point and its neighbors
- Loop through the robots and *Move* them one node at a time, updating controller's *exploredMap* with new explored nodes
- Track the number of "seen" nodes vs "visited" nodes
- Backtracking when no neighbor nodes to visit
- Stop mapping when "seen" nodes = "visited" nodes

Mapping Time Complexity

- $O(N + E^2)$ (worst case)
- $O(N + E)$ average case
- N is number of Nodes
- E is number of edges

Data Structures Used

- Graphs
- Vectors
- Stacks



- Robots are objects that traverse the graph node by node, communicating with controller to relay information about surrounding nodes and retrieve objects
- Controller handles the robots, sends/stores data from/to robots while traversing, also plans the paths for the robots to move

class Controller:

- vector<Robot*> robots
- vector<GraphNode*> visitedNodes
- Graph* graphMap
- Graph* exploredMap

class Robot:

- int robotID
- GraphNode* currNode
- stack<GraphNode*> prevNodeStack

HOURS SPENT PER WEEK

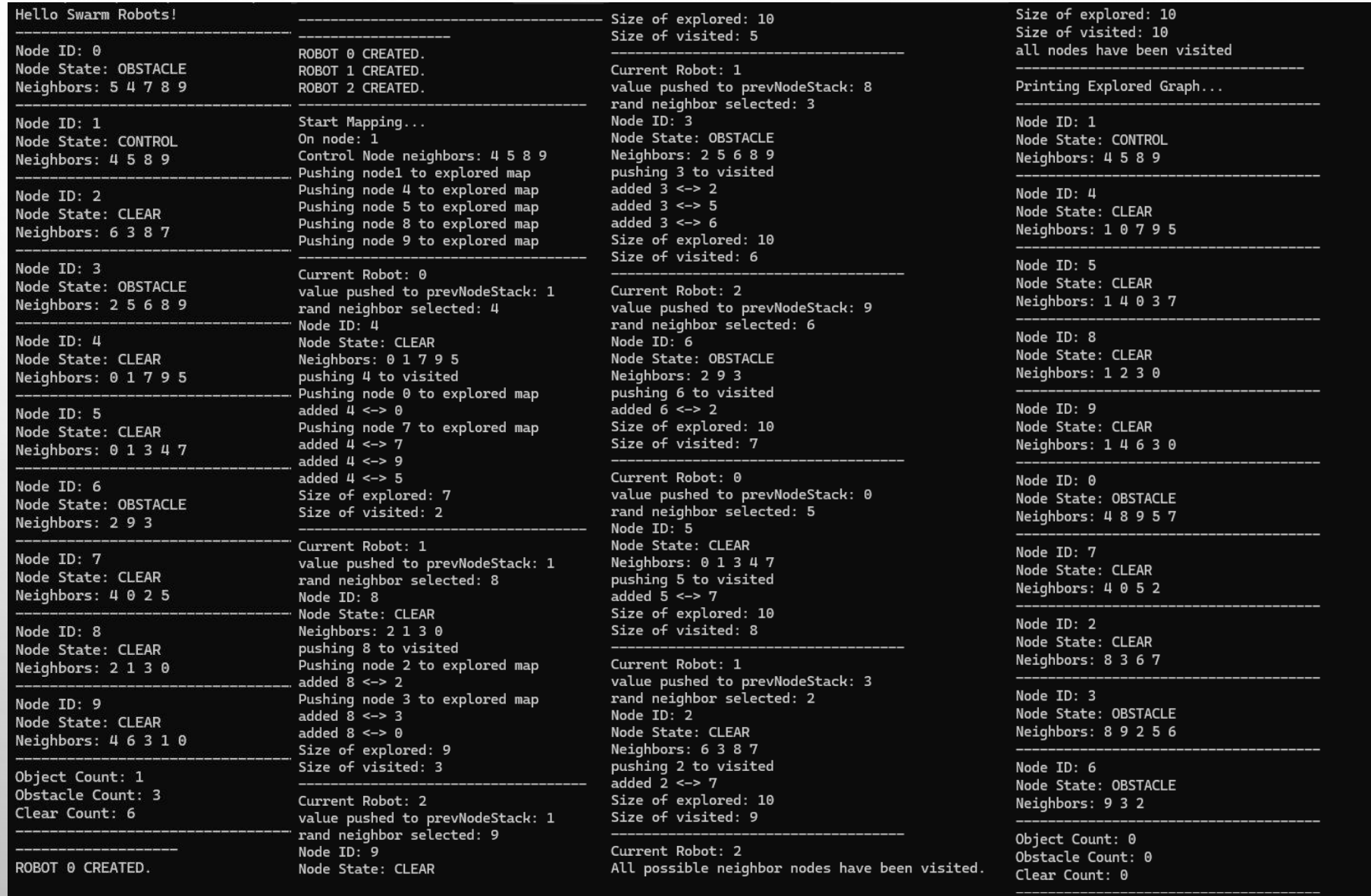
- 5 HOURS SPENT PER WEEK
- 4 WEEKS PER MONTH
- 20 HOURS PER MONTH

- GROUP CODING
- DISCUSSIONS

ANALYSIS AND RESULTS

RESULTS

- Created an algorithm to successfully create random graph environments
- Developed methods for robots to travel through graphs efficiently and ensuring they explore every node
- Created a platform for robots to map graphs completely to perform path planning



CONCLUSION

- Overall, gave idea of how professional software development teams operate
- Improved our understanding of graph theory
- Strengthened our problem solving and algorithm design skills

Projects Limitations/Need to Implement

- Have platform for path finding and object retrieval but still need to implement
- Going to use a modified Dijkstra's algorithm
- Implement obstacle node avoidance

Future Work

- Create a better interface for managing and analyzing swarm status
- Add different node states for more variability