RBE-550 Homework 2: Flatland Planning with BFS

Mohamed Eljahmi
Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts

***Abstract*—This report presents the implementation of a Flatland gridworld simulation for motion planning [3], [4]. The environment is modeled as a 64×64 occupancy grid with approximately 20% coverage generated from tetromino-shaped obstacles. A hero agent must navigate from a start position to a goal while avoiding ten greedy enemies. Enemies advance one step toward the hero at each time step and are eliminated upon colliding with obstacles, borders, or each other, leaving behind permanent junk obstacles. The hero replans its path at every step using Breadth-First Search (BFS) on a 4-connected grid. In a representative run, the hero reached the goal in 98 steps, with all ten enemies eliminated and 10 junk cells created. BFS expanded an average of 503 nodes per step, with expansions decreasing as the hero approached the goal. Simulation outputs include snapshots of the world, an animated run sequence, and performance plots.**

Keywords—Flatland; Motion planning; Gridworld; Breadth-First Search (BFS); Dynamic obstacles; Greedy enemies; Junk obstacles; Robotics simulation.

I. Introduction

Motion planning in dynamic environments is a fundamental problem in robotics. Robots must be able to navigate toward a target while avoiding static and dynamic obstacles. In this assignment, a simplified 2D gridworld called *Flatland* is used to study these challenges. The world is represented as a 64×64 grid with structured obstacles generated from tetromino shapes, which create approximately 20% occupancy of the grid.

The planning problem is defined as guiding a hero agent from a random start location to a randomly placed goal while avoiding ten adversarial enemies. Enemies move greedily toward the hero and are unaware of obstacles or each other. Collisions cause enemies to be removed from the game, leaving permanent junk obstacles. The hero has full knowledge of the world and must continually replan its path to the goal as the environment evolves.

Breadth-First Search (BFS) was selected as the planning algorithm for this assignment. BFS is complete and guarantees the shortest path in an unweighted grid, making it a suitable baseline for analysis. By replanning at every step, the hero can adapt to changes caused by enemy movement and junk accumulation. The simulation provides insight into the effectiveness of BFS in a dynamic world and highlights its limitations in terms of computational cost. The concept of configuration space [2] underlies this representation.

II. Methodology

A. Environment Setup

The world is represented as a 64×64 grid where each cell can be free, an obstacle, junk, or occupied by an agent. Obstacles are generated from tetromino shapes (I, O, T, L, S). Each tetromino is randomly rotated or flipped and placed at a random location if all cells fit within the grid and do not overlap existing obstacles. The process repeats until approximately 20% of the grid cells are filled. This structured approach produces clusters of obstacles rather than isolated blocked cells, resulting in more realistic navigation challenges.

B. Agent Placement

Three types of agents are placed on free cells: the goal, the hero, and ten enemies. Placement is random but ensures no overlap between agents and obstacles. The hero's objective is to reach the goal, while enemies pursue the hero.

C. Planning Algorithm

The hero uses Breadth-First Search (BFS) to plan its path. BFS expands nodes in a uniform manner on a 4-connected grid, guaranteeing the shortest path if one exists. At every time step, BFS is re-run to account for changes in the environment. The search treats obstacles, junk, and enemies as blocked cells. Additionally, a one-cell buffer (keep-out region) is enforced around enemies to reduce the risk of direct encounters. Each run logs the number of nodes expanded and the length of the path.

D. Enemy Model

Enemies move greedily toward the hero. At each time step, an enemy compares its row and column with the hero's and takes a step in the direction that reduces the Manhattan distance. Enemies ignore obstacles and each other when planning their moves. If an enemy collides with a wall, obstacle, or another enemy, it is eliminated, and its cell becomes junk. If an enemy steps into the hero's cell, the hero is caught, and the simulation ends in failure.

E. Simulation Loop

Each simulation begins with obstacle generation and agent placement. The hero computes its initial BFS path to the goal. At every step, the hero advances one cell along its current path, then enemies move. After enemies move, the BFS is re-run to update the hero's path. The process repeats until the hero reaches the goal, is caught by an enemy, or the maximum number of steps is reached. During each run, snapshots, a log file, and plots are generated to record performance and outcomes.

III. Results and Discussion

A representative simulation was performed on a 64×64 grid with an obstacle density of approximately 20% and ten enemy agents. The run was terminated successfully, with the hero reaching the goal in 98 steps. By the conclusion, all enemies had been eliminated, and 10 junk cells were created.

Snapshots of the hero's progress from steps 0 through 5 are shown in Figures 3–8. The initial configuration illustrates the random placement of tetromino obstacles and agents, while the final frame (Figure 9) confirms the hero's successful arrival at the goal. An animated sequence of the full run was also generated in GIF format (Figure 10).

Table I summarizes the performance metrics for this run. BFS expansions were recorded at each step, averaging approximately 503 nodes per step. The number of expanded nodes was highest during the initial stages and decreased steadily as the hero approached the goal (Figure 1).

The dynamics of enemy and junk evolution are presented in Figure 2. The enemy population declined rapidly during the first 18 steps, while junk cells accumulated to a stable total of 10 as collisions occurred. This outcome demonstrates how greedy enemy pursuit can inadvertently simplify the hero's planning task by progressively clearing the environment.

A. Simulation Setup

The experiments were conducted on a 64×64 grid with approximately 20% tetromino-shaped obstacles (Table I). The hero and goal positions were initialized at random free cells, while ten enemies were placed randomly throughout the environment. Simulation parameters are summarized in Table I.

Table I: Simulation Parameters (seed=42)

Parameter	Value
Grid size	64 x 64
Obstacle fill	~20%
Number of enemies	10
Seed	42
Max steps	200
Keep-out distance	1 cell
Steps to goal	98
Enemies eliminated	10 (all by ~step 18)
Final junk cells	10

B. Path Planning and Expansion

The hero successfully reached the goal in 98 steps. The BFS planner expanded to a maximum of ~1150 nodes at the start, with expansions monotonically decreasing as the hero approached the goal (Figure 1). This confirms BFS completeness [1], [4] while also showing its inefficiency on large grids due to exhaustive exploration.

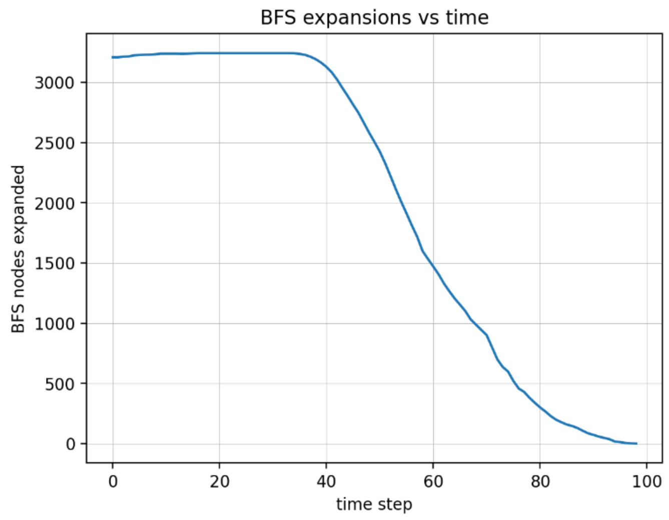

Figure 1. BFS node expansions per step. Average ~503 expansions per step across the run.

C. Enemy-Junk Dynamics

Enemy motion was purely greedy, often leading to collisions with obstacles or other enemies. As a result, the number of enemies rapidly decreased to zero by timestep 18, while the number of junk cells increased correspondingly to ten (Figure 2). This highlights how the environment evolves dynamically as adversaries remove themselves through collisions.

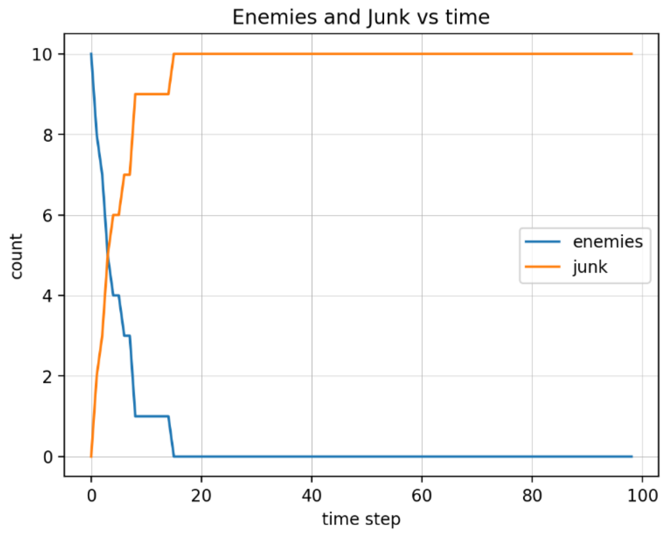

Figure 2. Enemy and junk count over time. Enemy population decreases while junk steadily increases.

D. Visual Snapshots of Simulation

Representative snapshots at key timesteps illustrate the interaction between hero navigation, obstacle distribution, and enemy dynamics (Figures 3–8). The hero consistently replanned paths to avoid both static and dynamic hazards.

- **Figures 3–8**: Step-wise world evolution
- **Figure 9**: Final state
- **Figure 10**: Animated sequence of the full run

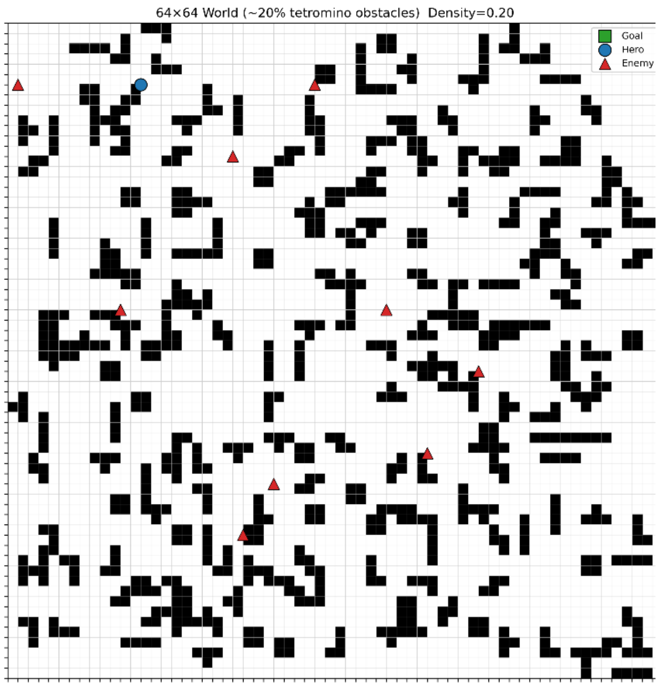

Figure 3. Initial world state (Step 0) showing tetromino obstacles, hero (blue), enemies (red), and goal (green)

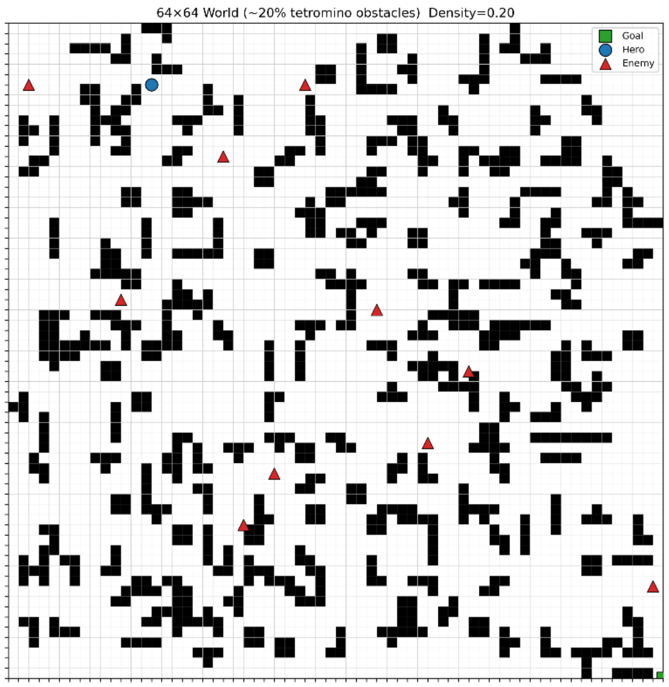

Figure 4. World state after Step 1. Enemies begin moving toward the hero.

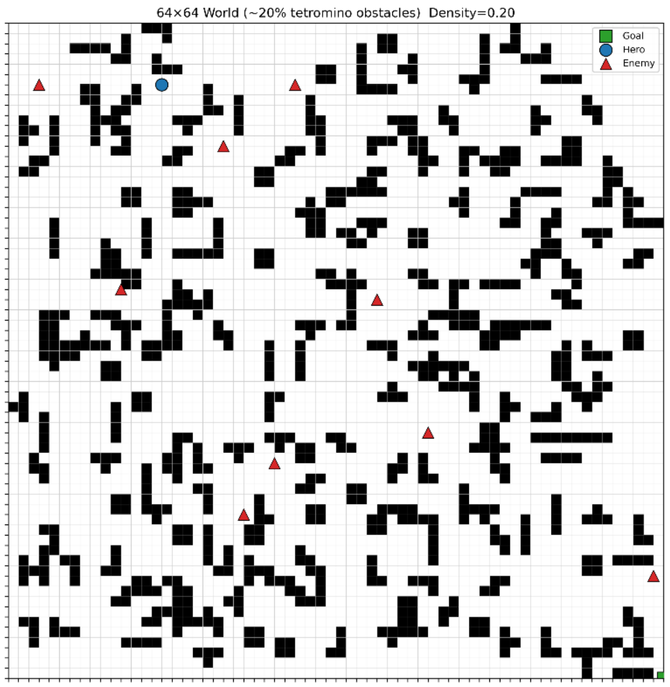

Figure 5. World state after Step 2. Enemy collisions generate junk cells.

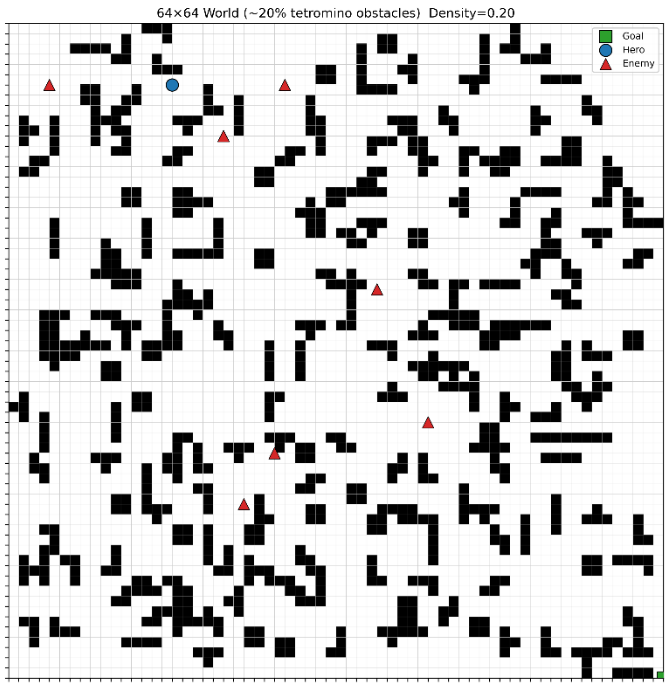

Figure 6. World state after Step 3. Enemy numbers decrease as junk accumulates.

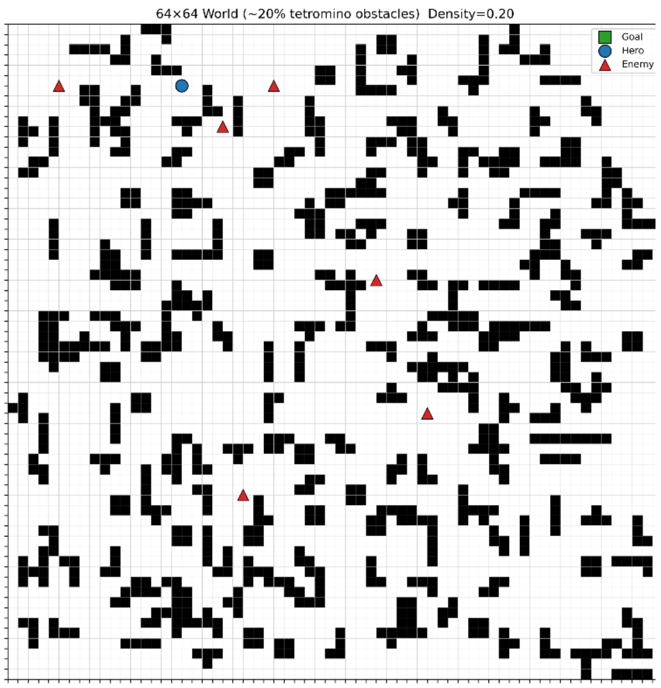

Figure 7. World state after Step 4. The hero advances toward the goal while enemies continue pursuit and junk cells appear.

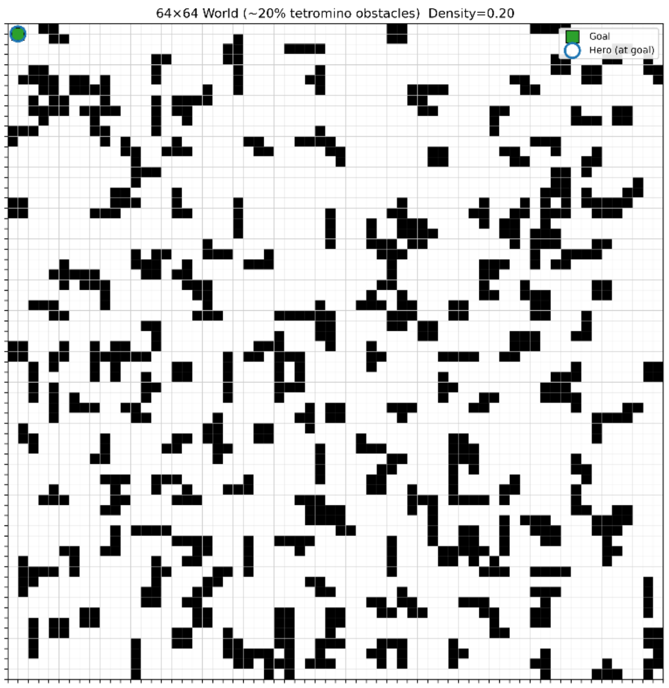

Figure 9. Final frame (Step 98) showing the hero successfully reaching the goal.

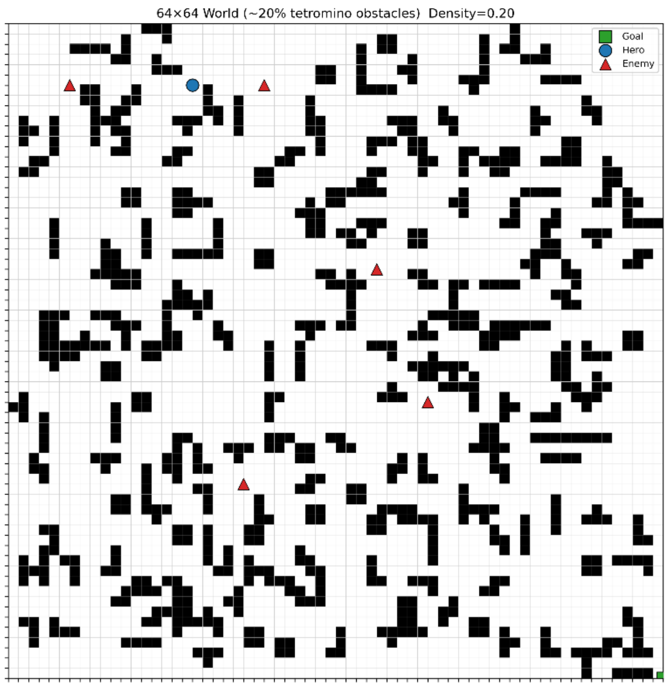

Figure 8. World state after Step 5. BFS path replans as the environment evolves with new junk and fewer enemies.

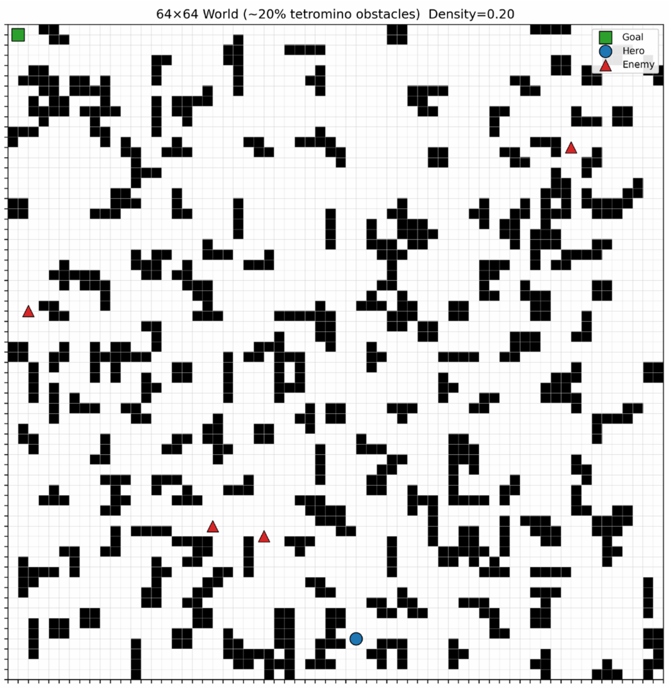

Figure 10. Animated sequence of the full run (GIF output, provided separately as a file).

E. Performance Metrics

From the run log, key metrics were extracted:

- **Steps to goal:** 98
- **Enemies eliminated:** 10 (all removed by timestep 18)
- **Final junk cells:** 10
- **Average BFS expansions per timestep:** 502.8

These results confirm the planner's effectiveness at reaching the goal while surviving in a dynamic, adversarial environment.

F. Discussion

The BFS approach guarantees completeness but incurs high expansion costs in the early timesteps. Although enemies decrease over time, they initially restrict safe regions, requiring the hero to replan frequently. The simulation demonstrates that environmental evolution (enemy-to-junk transitions) can indirectly aid the hero by clearing paths, though this is stochastic. A more efficient heuristic-based planner (e.g., A*) would likely reduce expansion significantly.

IV. Conclusion

This assignment implemented motion planning in a dynamic Flatland world using Breadth-First Search (BFS). The environment was constructed as a 64×64 grid with approximately 20% tetromino obstacles, a hero, a goal, and ten greedy enemies. The simulation demonstrated the ability of BFS to guarantee a valid path whenever one exists, while continually replanning to adapt to environmental changes caused by enemy pursuit and junk creation.

A representative run showed the hero successfully reaching the goal in 98 steps, with all enemies eliminated and 10 junk cells produced. BFS expansions averaged about 503 nodes per step, confirming the algorithm's completeness but also highlighting its computational inefficiency in large grids.

Overall, the Flatland simulation illustrates the strengths and limitations of BFS in dynamic environments. While BFS reliably guided the hero to the goal, its high expansion cost suggests the need for heuristic-based methods such as A* to improve scalability. Future work may also extend the model with teleportation, additional agent behaviors, or larger grid sizes to further test planning robustness.

References

[1] D. M. Flickinger, *Graph Search Algorithms*, RBE-550 Motion Planning Lecture Notes, Worcester Polytechnic Institute, Jan. 2023.

[2] D. M. Flickinger, *Configuration Space*, RBE-550 Motion Planning Lecture Notes, Worcester Polytechnic Institute, Jan. 2023.

[3] D. M. Flickinger, *Motion Planning for Point Robots*, RBE-550 Motion Planning Lecture Notes, Worcester Polytechnic Institute, Jan. 2023.

[4] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: http://lavalle.pl/planning/