Project Documentation

Contents

Introduction

Navigating environments with obstacles and constraints is a crucial challenge in robotics applications. Finding optimal paths for robots to move efficiently and safely requires robust algorithms that can handle diverse scenarios. This document presents a genetic algorithm (GA) approach to address this challenge.

Problem Definition:

Input:

- Environment Map: Represented as a grid with obstacles, where each cell can be either occupied or free.
- Start and Goal Positions: Initial and final positions where the robot starts and needs to reach.

Output:

- Optimal Path: A sequence of movements or actions that the robot can take to reach the goal while avoiding obstacles and following the constraints.

Challenges:

- Obstacle Avoidance: The robot needs to navigate around obstacles to reach the goal.
- Optimization: Finding the shortest or most efficient path based on defined criteria (e.g., distance traveled, time taken).
- Constraint Adherence: Ensuring that the robot respects limitations such as speed, turning capabilities, or any other imposed restrictions.
- Real-time Planning: For dynamic environments, the path planning might need to be adaptable and quick.

Genetic Algorithm Implementation :

Individual Class:

- Description: Represents an individual path as a sequence of movements for the robot within the environment
- Attributes:
  - **genes**: Represents the sequence of moves as a list of tuples (e.g., **(1, 0)** for moving right, **(0, -1)** for moving down).
- Methods:
  - **__init__(self, genes=None)**: Initializes an individual with provided genes or an empty list if not specified.

Obstacle Class:

- Description: Defines obstacles within the environment by specifying their position and dimensions.
- Attributes:
  - x: x-coordinate of the obstacle's position.
  - y: y-coordinate of the obstacle's position.
  - width: Width of the obstacle.
  - height: Height of the obstacle.
- Methods:
  - __init__(self, x, y, width, height): Initializes an obstacle with specified position and dimension

Graph Class:

- Description: Represents the environment in which the robot operates, including dimensions, start and end points, and randomly generated obstacles.
- Attributes:
  - width: Width of the environment.
  - height: Height of the environment.
  - start: Starting point for the robot's path.
  - end: Ending goal point for the robot's path.
  - obstacles: List of obstacles within the environment.

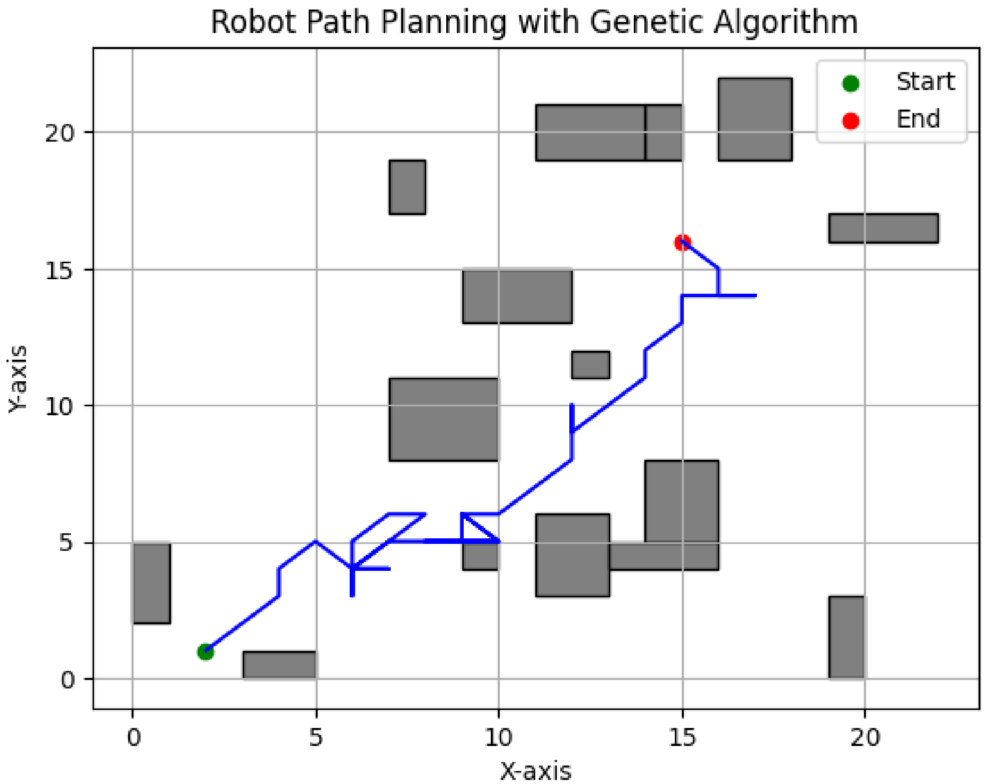
- Methods:
  - generate_random_obstacles(self, num_obstacles): Generates a specified number of random obstacles within the environment.
  - is_valid_move(self, x, y): Checks if a given move is valid, considering obstacles and boundaries.

GeneticAlgorithm Class:

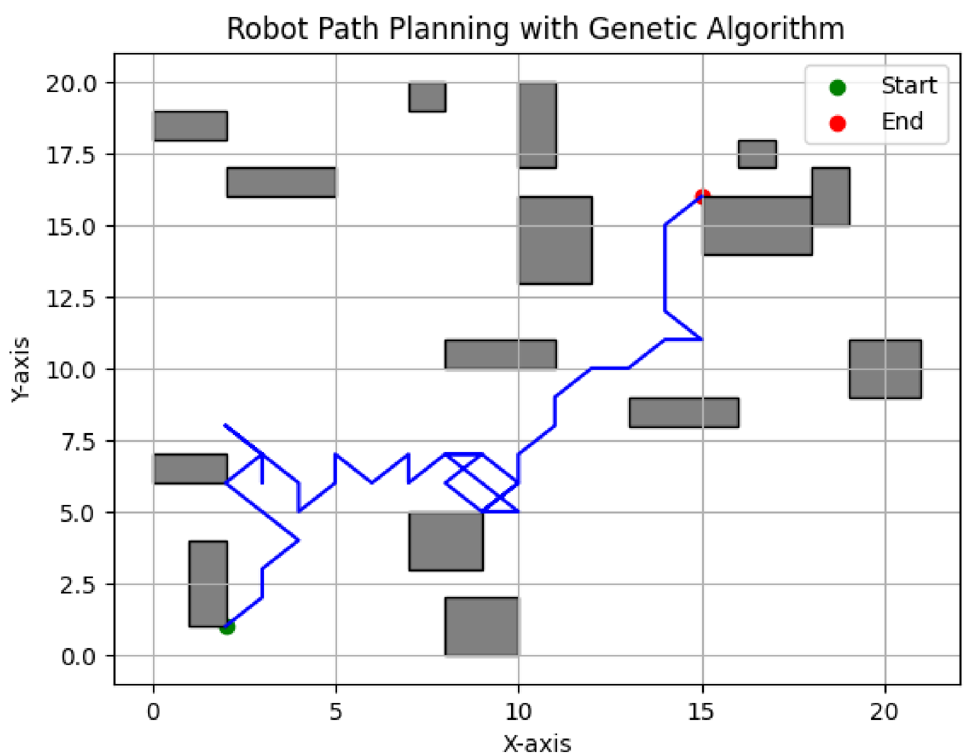
- Description: Implements the genetic algorithm for robot path planning within the defined environment.
- Attributes:
  - population_size: Size of the population for genetic algorithm operations.
  - graph: The environment represented as a graph.
  - start: Starting point for the robot's path.
  - end: Ending goal point for the robot's path.
  - iterations: Number of iterations for the genetic algorithm.
  - population: List representing the current population of individuals (possible paths).
- Methods:
  - generate_random_individual(self): Generates a random individual (path) for the population.
  - initialize_population(self): Initializes the population with random individuals.
  - fitness_function(self, individual): Evaluates the fitness of an individual path.
  - select_parents(self): Selects individuals from the population for reproduction based on fitness.
  - crossover(self, parent1, parent2): Performs crossover operation to create offspring from parents.
  - mutate(self, individual): Introduces random changes to an individual's path while ensuring validity.
  - evolve(self): Drives the evolution process of the population through multiple iterations.
  - plot_solution(self, best_individual): Visualizes the best path found by the genetic algorithm.

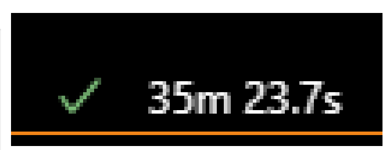
Results, Analysis and Appendix:

population size	100	iterations	500

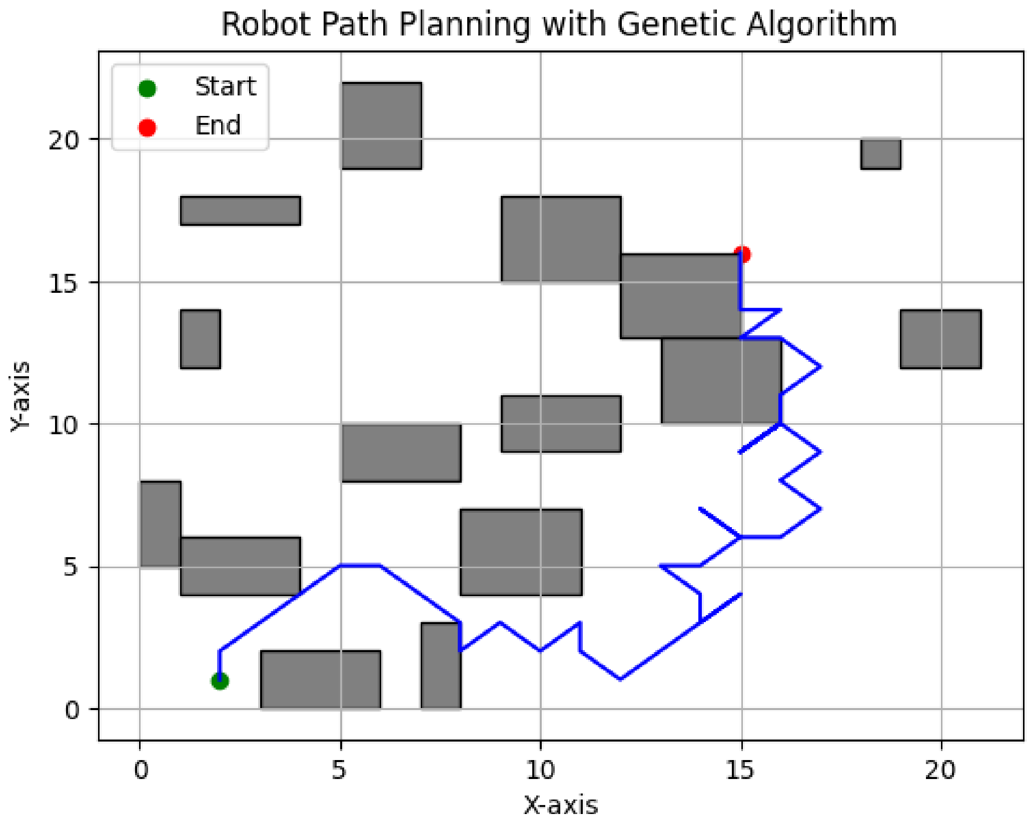


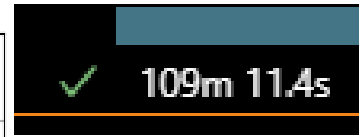
population size	600	iterations	50



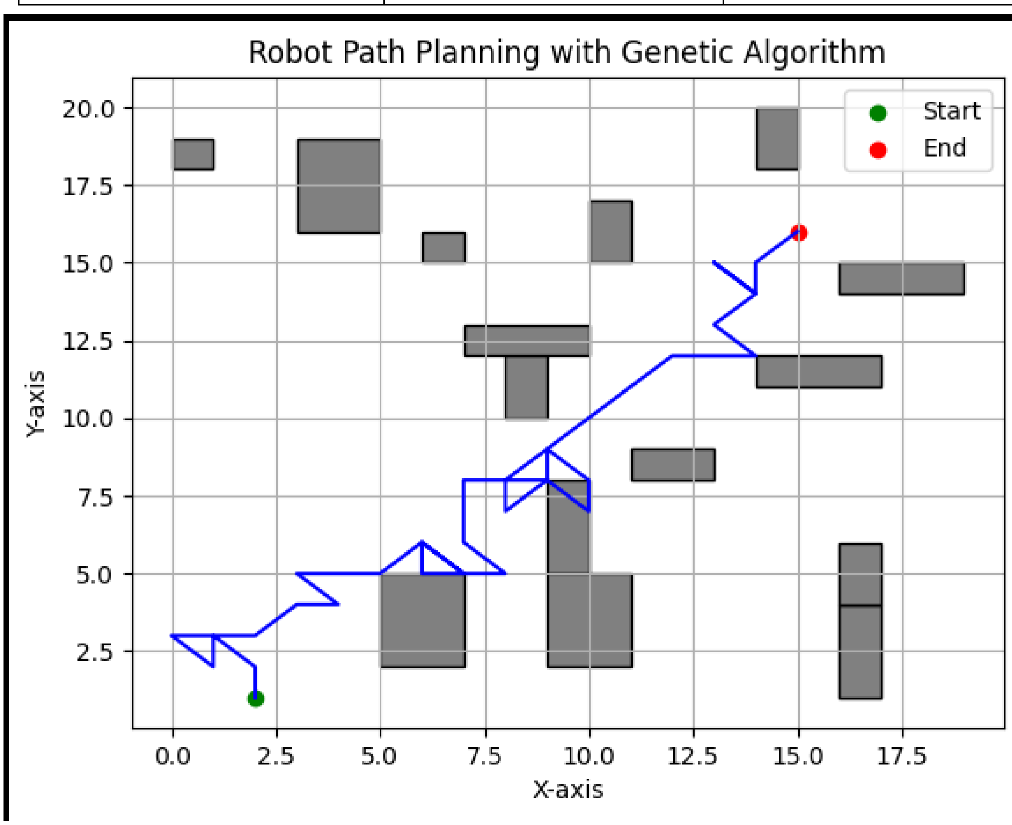


population size	100	iterations	5000

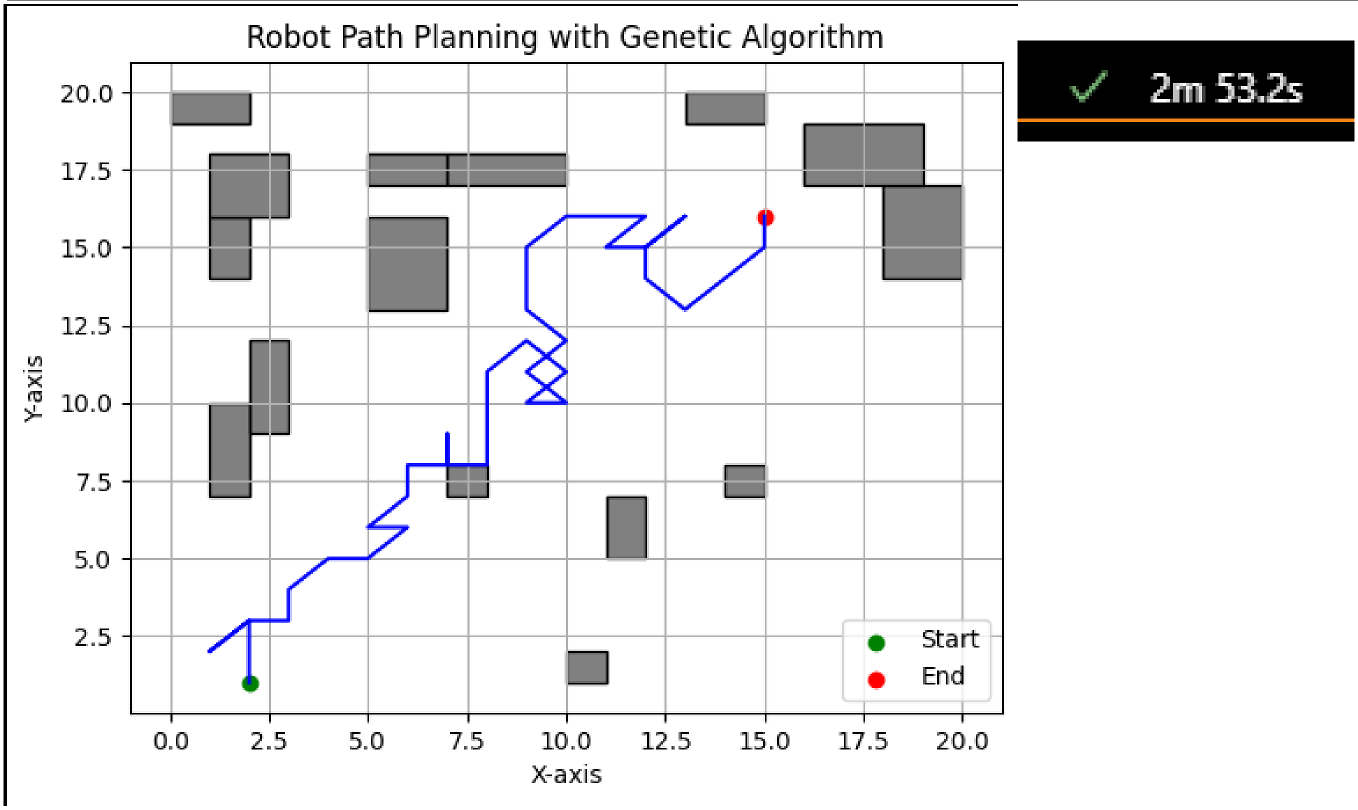




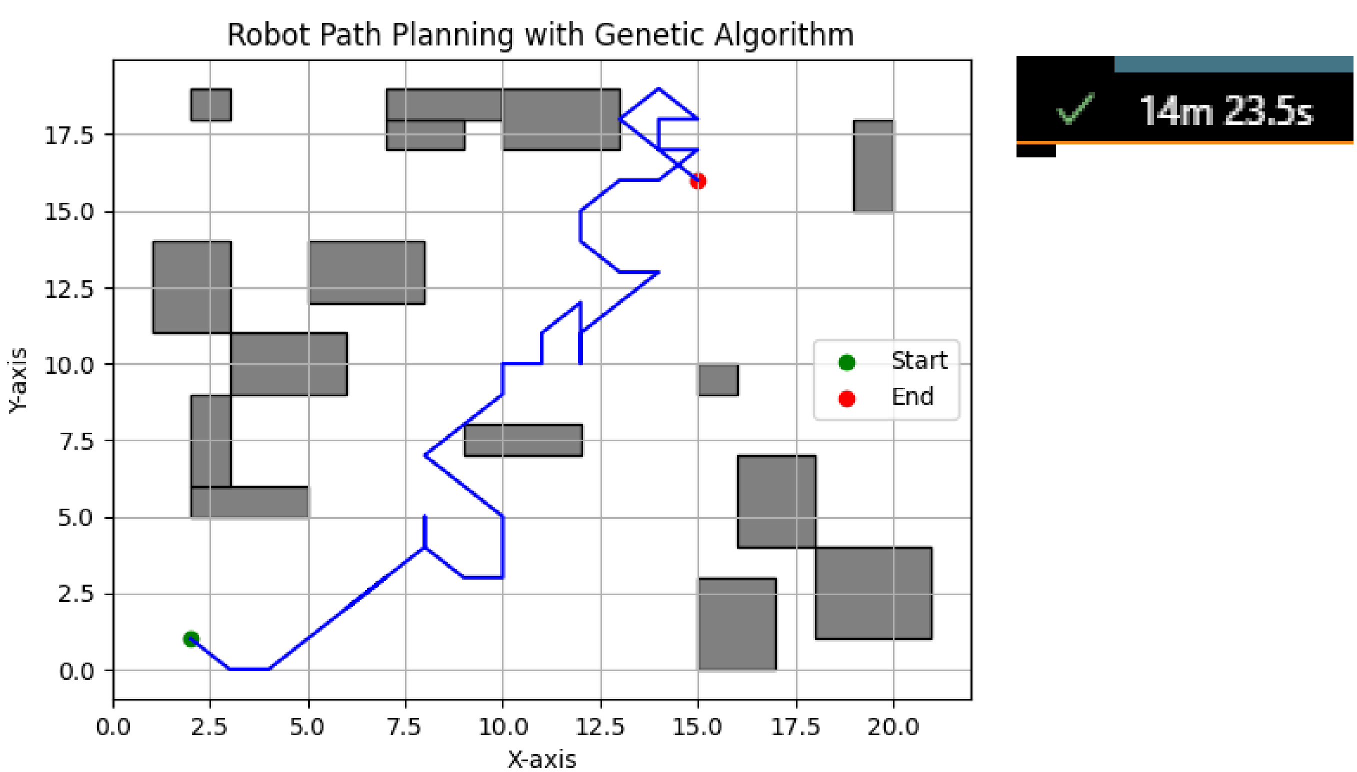
population size	50	iterations	200



population size	50	iterations	500



population size	30	iterations	5000



conclusion:

The application of genetic algorithms to tackle robot path planning in obstacle-laden environments manifests as a promising avenue in robotics. Key insights gleaned from this approach include:

- Algorithmic Efficacy: Genetic algorithms offer a systematic and efficient means to navigate complex environments while finding optimal paths.
- Class Structures' Harmony: The synergy among Individual, Obstacle, Graph, and GeneticAlgorithm classes fosters a cohesive environment for simulating and determining optimal robot paths.
- Flexibility and Adaptability: This approach accommodates various environmental configurations and constraints, displaying adaptability in dynamically changing scenarios.

In essence, leveraging genetic algorithms in robot path planning furnishes a robust methodology that empowers robots to navigate intricate environments adeptly, fostering efficiency and safety in their operations.