Collision-Free Shortest Path Planning for Robots using Genetic Algorithms

By Adric Crawford, Vincent Grady and Michael Aguadze.

Department of Electrical and Computer Engineering

North Carolina Agricultural and Technical State University

People

- Team
  - Adric Crawford, UG Scholar, Department of Electrical and Computer Engineering
  - Vincent Grady, UG Scholar, Department of Electrical and Computer Engineering
  - Michael Aguadze, PhD Scholar, Department of Electrical and Computer Engineering
- Submitted to:
  - Dr. Abdollah Homaifar
- Inspiration:
  - Dr. Kamal Azmyin

Content

- Problem Definition
- Introduction to Mobile Robots
- Encoding Schemes: Binary vs. Integer
- Chromosome Length
- Population Size and Initialization
- Binary Encoding Process
- Key Features
- Code Implementation
- Design Variable Parameters
- References

Problem Definition

Given the forward simulated states of neighboring vehicles, predicted trajectories, and environment of ego vehicle, find the best feasible trajectory for the ego vehicle using a genetic algorithm.

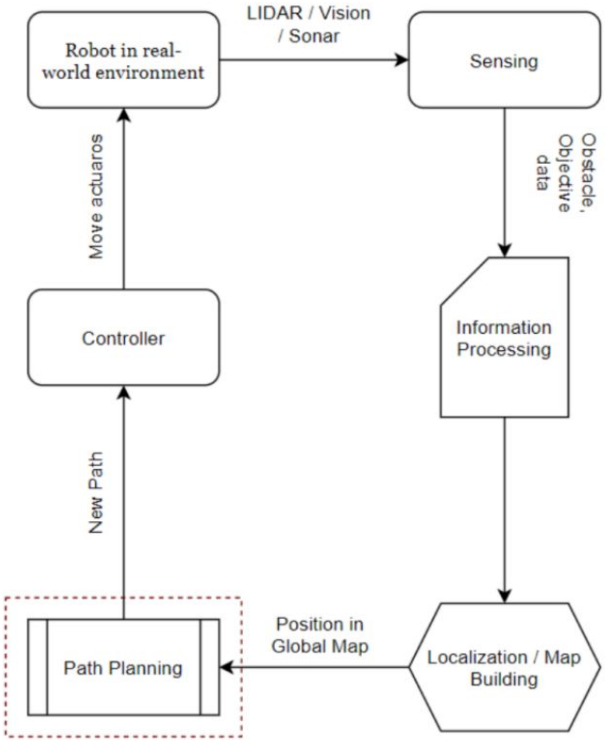


Figure 1: General architecture of a mobile robot

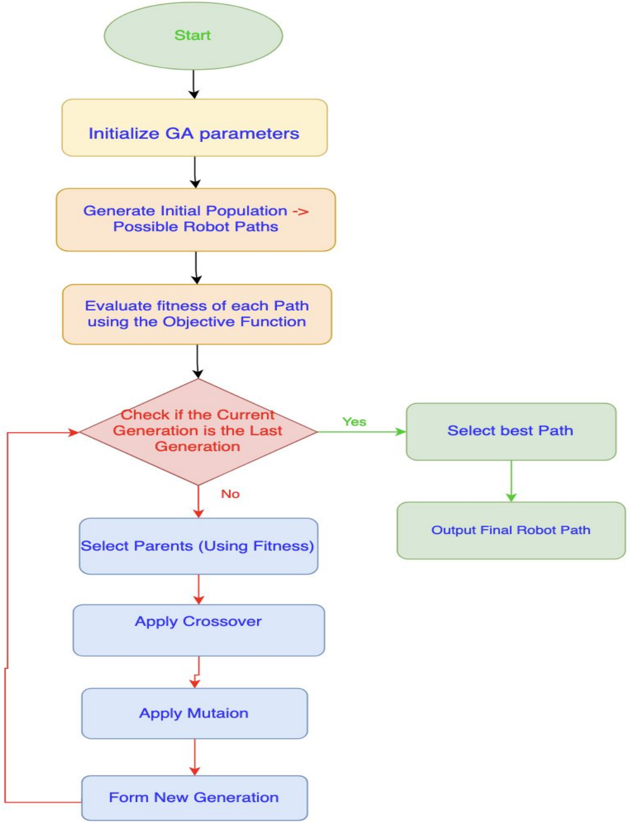
Introduction to Mobile Robots

- A mobile robot is an intelligent vehicle that:
  - » Perceives the workspace.
  - » Acquires and interprets sensory data.
  - » Determines its location.
  - » Formulates a motion plan.
- Objectives:
  - » Least energy movement.
  - » Collision-free navigation.
  - » Coordinated movement.
- Applications:
  - » Medicine, entertainment, agriculture, mining, rescue, education,
  - » military.

How the Vehicle Uses the GA

- Objective to rapidly and efficiently determine the best feasible path through obstacles for dynamic settings with local path-planning.
  - Uses fewer computer resources than methods like A*, Road Map
- Vehicle starts with a known global path
- New maps are generated from sensor data on the local environment
  - These maps are quickly evaluated with the GA
  - Vehicle constantly adjusts path based on final result of the new algorithm
  - The map does not change while the algorithm is running

Algorithm Flow Chart



Important Concepts

Encoding Schemes: Binary vs. Integer

- Fundamental design decision: Encoding scheme.
- Options:
  - Binary Encoding
  - Direct Integer String Operations
- Binary Encoding (Adopted Approach):
  - Integer string → Binary string.
  - GA operations (roulette wheel, crossover, mutation).
  - Binary string → Integer string.
  - Conforms to literature’s techniques.
  - Binary representation = Chromosome.

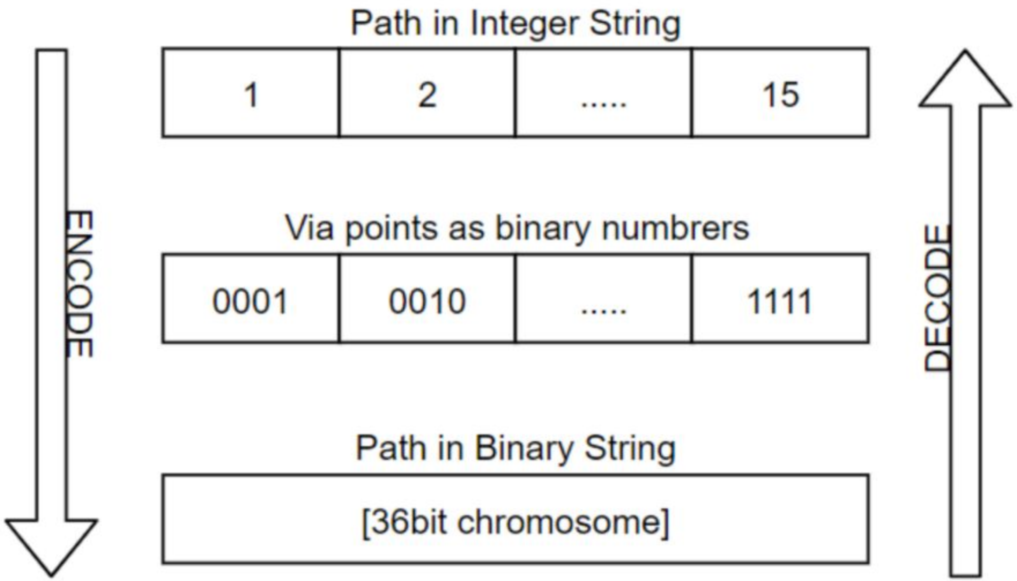
Chromosome Length

- Chromosome length: Fixed or variable.
- 4-bit representation: 16 possible via points.
- Maximum chromosome length: (16 * 4) = 64 bits.
- Fixed length choice: No consensus.
- Nagib et al. approach (Adopted):
  - » Length = (m + 2) * n bits
  - » m: Number of static obstacles.
  - » n bits: Bits per via point.

Population Size and Initialization

- Population size: Fixed.
- Classical approach: Random initial generation.
- Adopted approach: Insert known paths into initial population.
- Motivation:
  - Without known paths, GA fails or finds poor solutions.
  - Limited generations (20-50).

Binary Encoding Process



Objective/Fitness Function(s)

- Objective: Maximize f(X)
- Function:

$$f(x) = \begin{cases} \dfrac{1}{\sum_{i=1}^{m+1} d(P_i, P_{i+1})} : Feasible\ path \\ 0.001 : Infeasible\ Path \end{cases}$$

$$d(P_i, P_{i+1}) = \sqrt{(X_{i+1} - X_1)^2 + (Y_{i+1} - Y_1)^2}$$

- d(Pi, Pi+1): Euclidean distance between two points.
- Workspace 1: [1 - 2 - 5 - 6 - 7 - 15 - 15 - 15] → F\f(x) = 50.72
- Workspace 2: [1 - 5 - 7 - 8 - 10 - 15 - 15 - 15] → f(x) = 55.71
- Considering modification to this function for best feasible paths (more later)

Key Features

- Two-point crossover between the first 4 and last 4 bits
- Bitflip mutation
- Roulette Wheel Selection
- 4 bits to represent points
- Two new heuristic rules

Design Variable & Parameters

- Design Variable: Sequential integer string (e.g., 1-2-3-4-5-6-7-9-15)
- Parameters:
- s loc = 1, e loc = 15
- N: Number of candidates
- num iter: Number of iterations
- known solution = 4
- Nx: Number of GA executions

```
% HARDCODED, CHANGES WITH MAP OF THE ENVIRONM
bit_count = 4; % [4 = paper, 8 = experimental
m = 7; % Number of static obstacles,
% HARDCODED, CHANGES WITH MAP OF THE ENVIRONM

% Provide inputs
N = 10; % Number of candiates per generation
num_iter = 75; % How many generations to try

% HARDCODED, if you change this, you must pro
% random_g1.m
s_loc = 1;
e_loc = 15;

% Input, starting location and finishing loca
[point_mat, path_index, point_ls] = load_dat(
% point_ls = [start_point,finish_point,min_po
```

Room For Improvement

Alternative Selection Processes

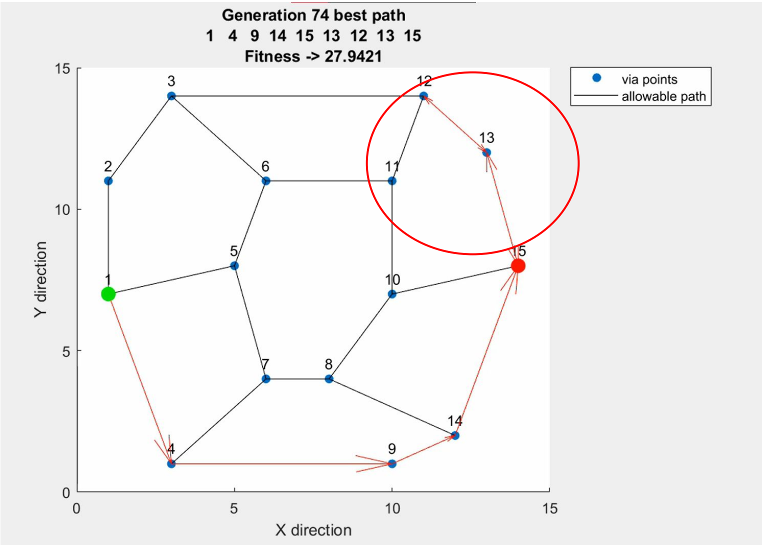
- The current selection process requires global elitism to maximize odds of retaining the best feasible solution of every generation.
- Other reproduction strategies may alleviate this such as
  - Tournament selection
    - Higher selection pressure
  - Rank selection
    - Controlled selection pressure
- Alternative selection methods will be explored in the future without global elitism to determine their effect on its necessity

Discussion on Suboptimal Pathing

- As shown in the plots, suboptimal paths are produced in both maps. This is due primarily to a few factors.
  - The inherent pseudo-randomness of genetic algorithms
  - Suboptimal population size and or number of generations
- Possible solutions:
  - Run the program multiple times to improve odds of producing best possible results
  - This is when the best possible path is know. But wouldn't be if possible path is unknown
  - Increase number of generations

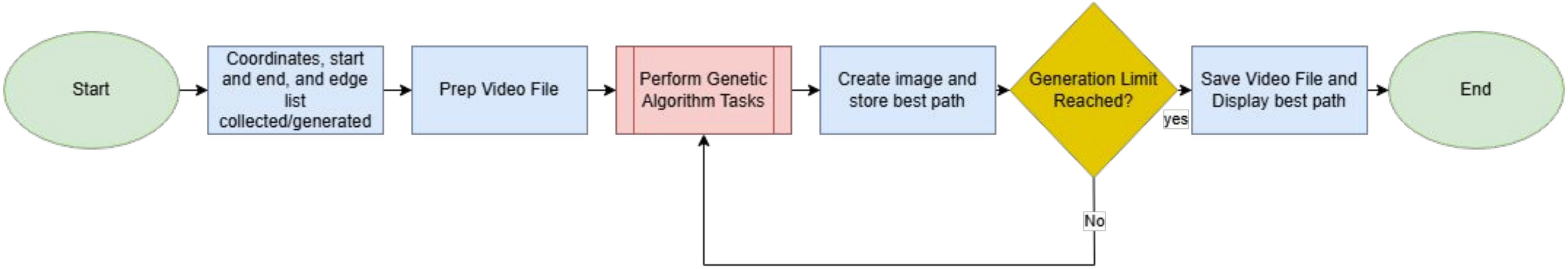
Dynamic Path Length for Optimization

- Static string size requires some back tracking for best feasible solutions
- Possible remedy:
  - » Evaluate fitness based on the first time the string reaches the end not the last
  - » Ex [1,4,9,14,15,13,12,13,15]
    - Evaluate the fitness of the path presented by the first five numbers in this instance



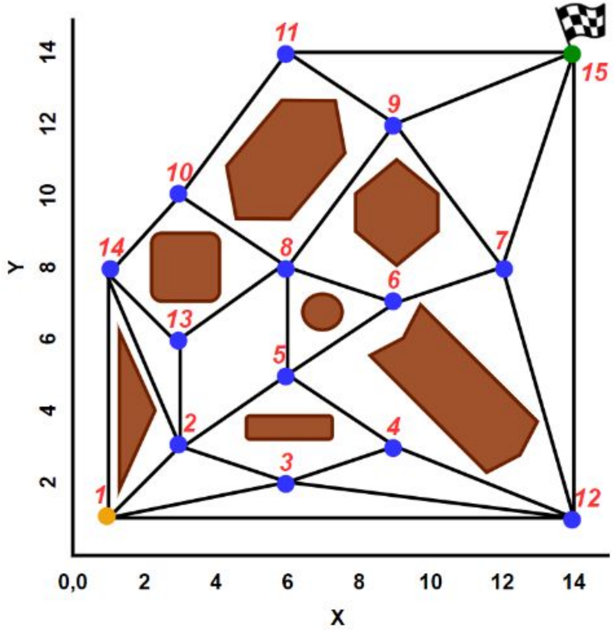
How the Code Works

High Level Code Flow Chart



Generating Maps

- In the intended setting, there will be a static global map generated beforehand of the expected environment.
  - Afterwards, new maps are generated and processed rapidly from sensor data
- For using the code, a map can be hardcoded and generated in load_dat.m using coordinates

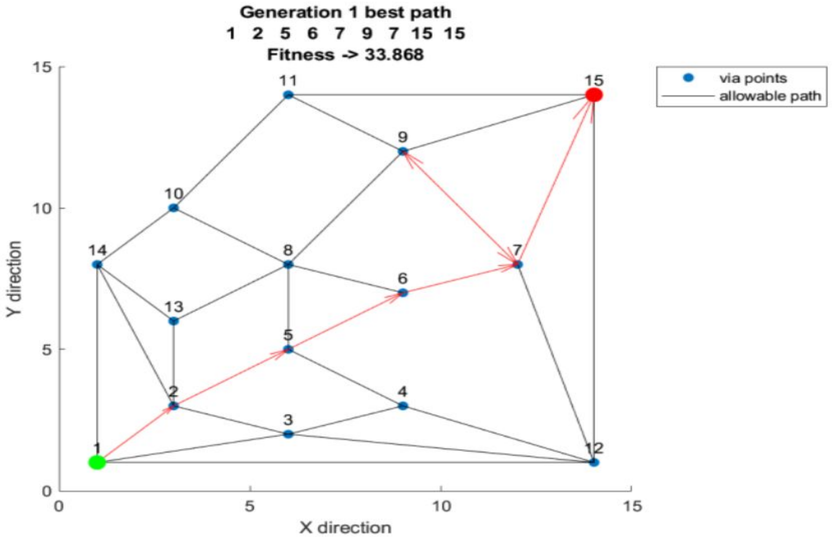


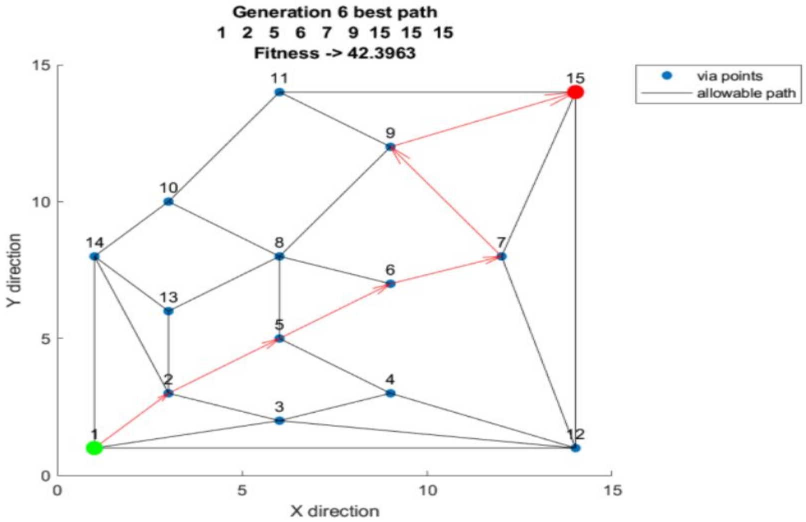
Ensuring Best Possible Results

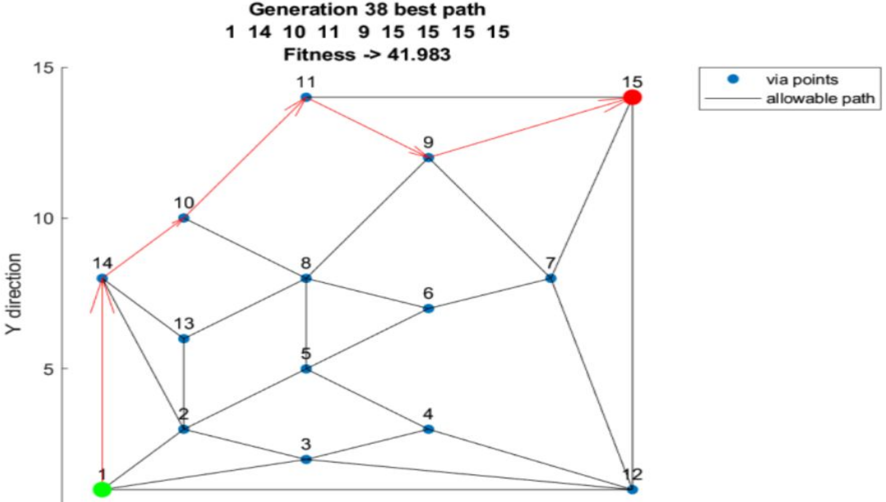
- The nature of genetic algorithms introduce some issues like premature convergence
- The program resets the current generation if too many individuals are identical to prevent premature convergence
- This algorithm has a way to inject the most fit solution found into every single generation to ensure it is not lost due to randomness

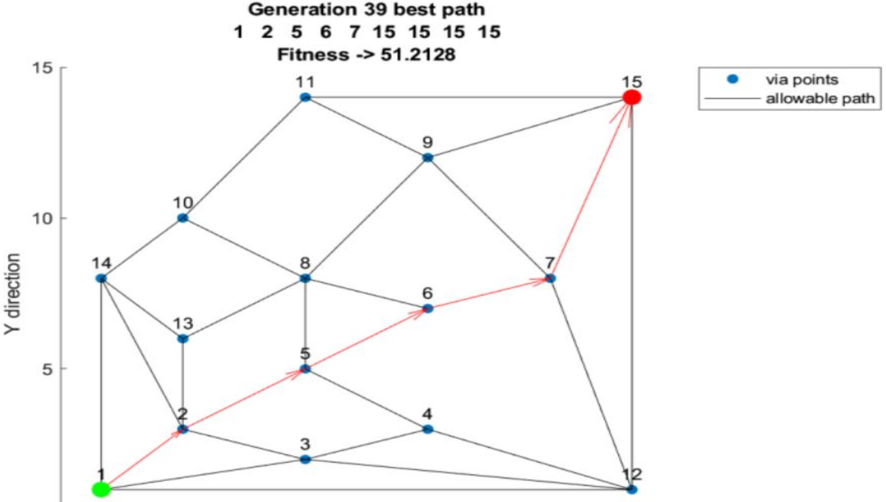
Table 7: Results for Experiment in Workspace 1

Exp No.	iter_no	N	Best path	fitness value [0 ~ 100]	Found known best path ?
1	50	10	[1 2 5 6 7 9 15 15 15]	42.39	No
2	50	10	[1 2 5 6 7 9 15 15 15]	42.39	No
3	50	10	[1 2 5 6 7 15 15 15 15]	51.73	Yes
4	50	10	[1 2 5 6 7 9 15 15 15]	42.39	No
5	50	10	[1 2 5 6 7 9 15 15 15]	42.39	No
6	50	10	[1 2 5 6 7 9 15 15 15]	42.39	No
7	50	10	[1 2 5 6 7 9 15 15 15]	42.39	No


Generation 1 best path
1 2 5 6 7 9 7 15 15
Fitness -> 33.868
via points
allowable path
Y direction
X direction


Generation 6 best path
1 2 5 6 7 9 15 15 15
Fitness -> 42.3963
via points
allowable path
Y direction
X direction

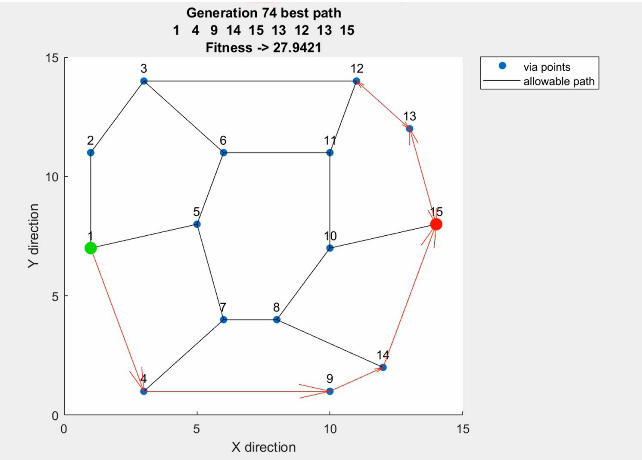

Generation 38 best path
1 14 10 11 9 15 15 15 15
Fitness -> 41.983
via points
allowable path
Y direction


Generation 39 best path
1 2 5 6 7 15 15 15 15
Fitness -> 51.2128
via points
allowable path
Y direction

Path Results

- Parameters:
  - » Bit_count=4
  - » m=7 (obstacles)
  - » N=10 (population)
  - » Generations = 75
  - » start/end=[1,15]
  - » Map: Pre gen 1

- Bottom image shows similarity prevention disabled
  - » Note how the global elite is present but the rest of the strings are stuck



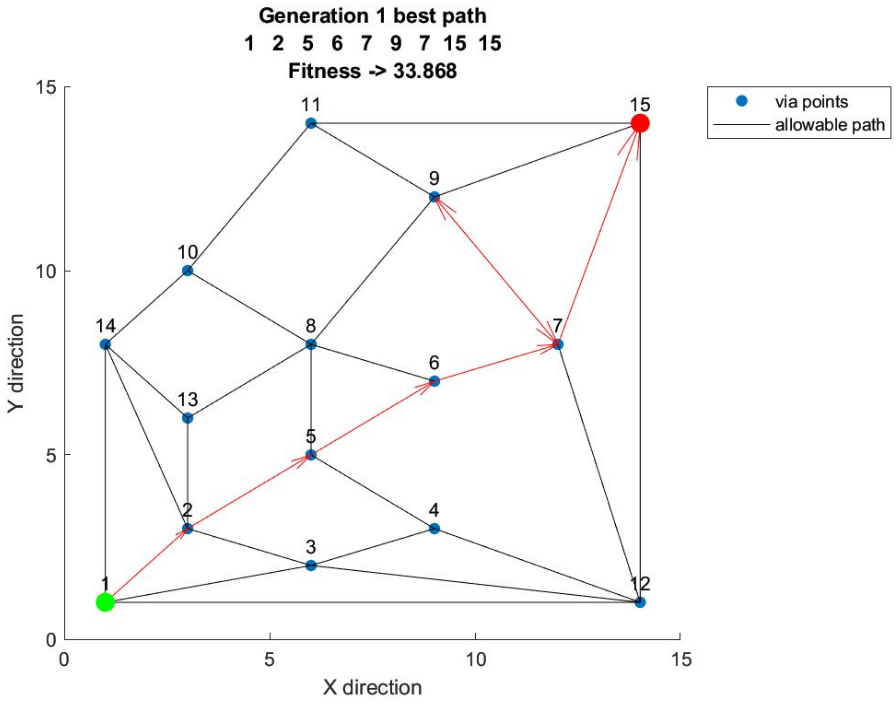

```
--------------------- Candiates fitness---------------
ans = 1×10
```

	1	2	3	4	5	6	7	8	9	10
1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	27.9421	27.5515	27.9421

```
--------------------- Candiates fitness---------------

--------------------- Candiates fitness---------------
ans = 1×10
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000   27.9421

--------------------- Candiates fitness---------------

--------------------- Candiates fitness---------------
ans = 1×10
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000   27.9421

--------------------- Candiates fitness---------------
```

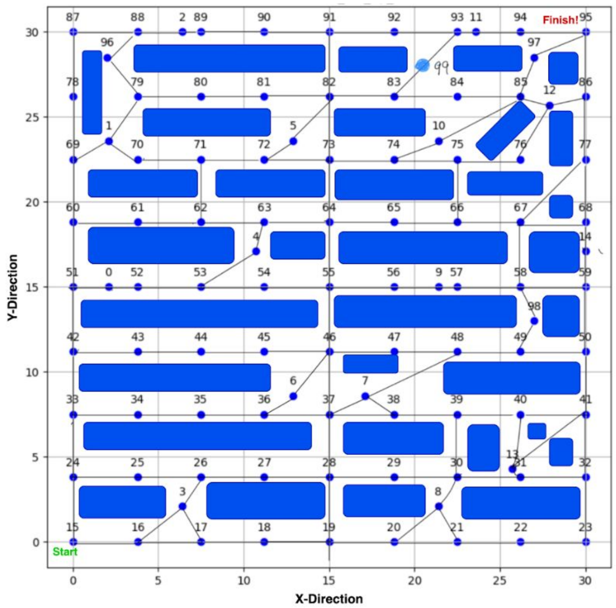
Path Results

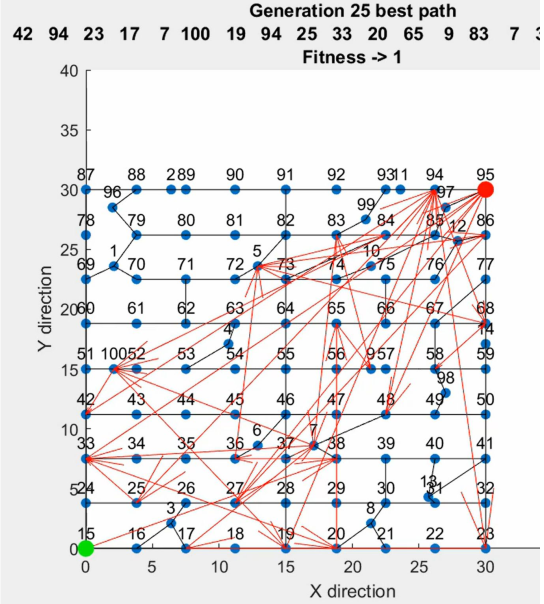
- Parameters:
  - Bit_count=4
  - m=6 (obstacles)
  - N=10 (population)
  - Generations = 100
  - start/end=[1,15]
  - Map: Pre gen 2



Path Results

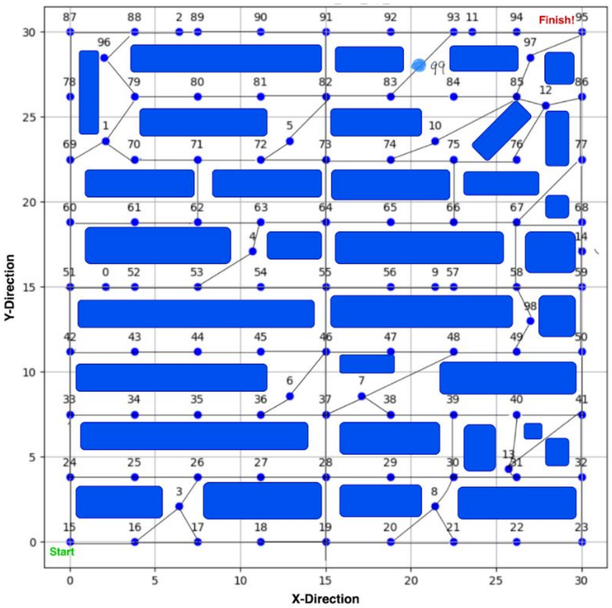
- Parameters:
  - Bit_count=7
  - m=32 (obstacles)
  - N=20 (population)
  - Generations = 2
  - start/end=[15,95]
  - Map: Pre gen 4

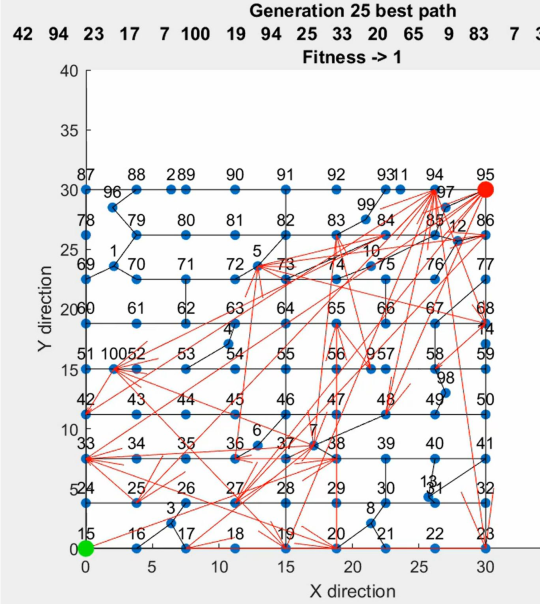




Conclusion

The implementation of a Genetic Algorithm for mobile robot path planning successfully demonstrated viability in small, structured environments with clearly defined nodes and obstacles. However, the algorithm's performance declined significantly when applied to larger-scale maps, where larger chromosome length.





References

- Ding, W., Zhang, L., Chen, J. and Shen, S., 2019. Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor. IEEE Robotics and Automation Letters, 4(3), pp.2997-3004.
- Kamal, Azmyin Md. Collision-Free, Shortest-Path Planning for Mobile Robots in 2D Static Workspace using Genetic Algorithm. Technical report, North Carolina A&T State University, submitted in partial fulfillment of ENGR 635, 2025.

Questions?