2015-16

Term Project on Application of Genetic Algorithm

**Topic: Path Planning using Genetic Algorithms**

Course: MT21104 -**Genetic Algorithms in Engineering Process Modeling**

1. Partha Sarathi Mishra [14MF3IM08]
2. Patil Amit Sanjay [14ME30030]
3. Shubham Dhal [14ME10078]

Presented by:

Abstract:

This Paper presents use of Genetic Algorithms for Path Planning in Mobile Robots. A Genetic Algorithm is used to find Optimal Path for robot in Map (Static) with nodes and links. Each Location is seen as Landmark (point) in 2-D Space. Each location in map is Represented as a gene which is binary Coded. Landmarks of Target and Obstacle are given in Map or 2-D space in which we are working. A chromosome consists of number of genes depending upon number of obstacles/points in Map. The Robot has a starting Point and a target point assuming the robot passes each point only once or not. The Obtained Result shows the Potential of Proposed algorithm.

Introduction:

The Path Planning Problem in brief can be Presented as => Given data: (1. Starting Location 2. Goal Location 3. 2-D Workspace consisting node locations and connectivity), Plan a collision free trajectory between two specified points satisfying criteria for optimal path or Shortest path. The Path Planning Problem is computationally very expensive. In spite of a great research being done to address this problem, conventional approaches become infeasible due to

* Change in Optimization Goals.
* Limits of Computational Resources.

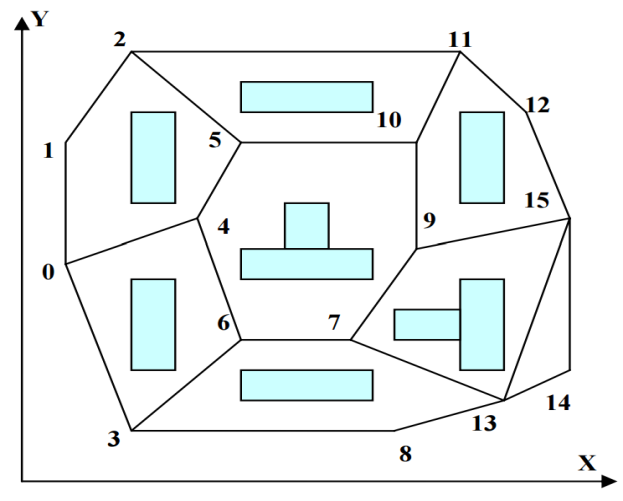
Hence by using Genetic Algorithm we have been able to present a solution which is both flexible and scalable.

Problem Formulation:

The Area Described by 2-D Workspace is the region Allowed for mobile robot to move. This 2-D map includes Several obstacles which are Provided at Runtime. These obstacles are represented by set of points which allows the mobile robot to avoid them. The Starting and Destination points are also Provided. The main Objective is to plan a collision free path which is also the optimal(shortest) path between the source and destination. The following design objectives are considered while finding the optimal path :-

* Minimize Distance travelled Between Source and Destination.
* Travel through one point only once.
* Reject Infeasible paths

In solving this problem we have assumed that the map of the location with the position and size of the obstacles has already been calculated and provided to us in the form of points through which the robot can move. The points and the interconnecting paths specify the permissible paths for the mobile robot .



Mapping the Workspace to find permissible points/paths

Procedure and Algorithm :

As stated earlier the code used takes the position(co-ordinates) and connectivity of the points as inputs . After which one can specify the Start and End points upon which code will specify the shortest feasible path between those two points.

First we started off by taking the co-ordinates and connectivity of the nodes as input . These were stored for easy access in subsequent steps . Then the code generates a random population of strings specifying the path co-ordinates and assigns fitness to the individuals.

Fitness is defined as 1/Path Distance for feasible paths and 0 for infeasible paths .

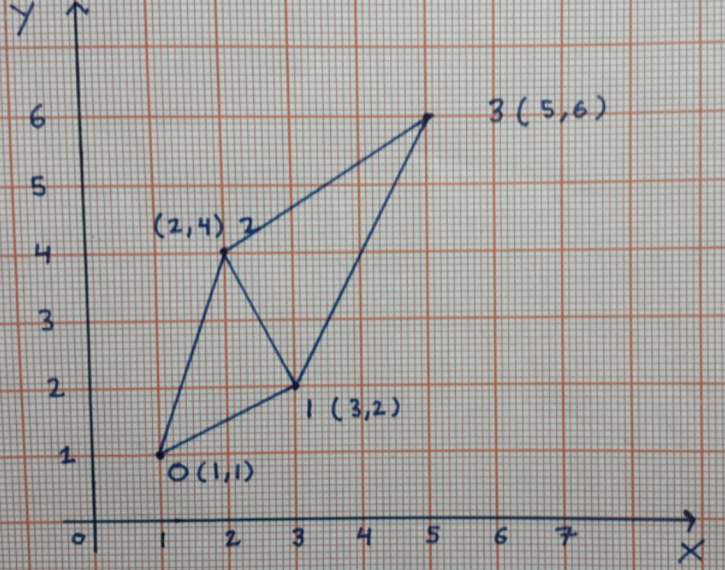
Selection and Crossover is carried on the current generation to find the next generation . We didn't consider mutation while finding the next generation .The Selection operator considered is quite simple , wherein we drop the least fit individual and make 2 copies of the fittest individual . We have considered single point crossover while evaluating the next generation.

The code is written in python and has been included in a separate document for reference .

Simulation and Results :-

We did some simulations for some simple cases with limited number of nodes .

1.



Number of Nodes : 4 (0,1,2,3)

Starting Node : 0

Ending Node : 3

Population after 1000 iterations :

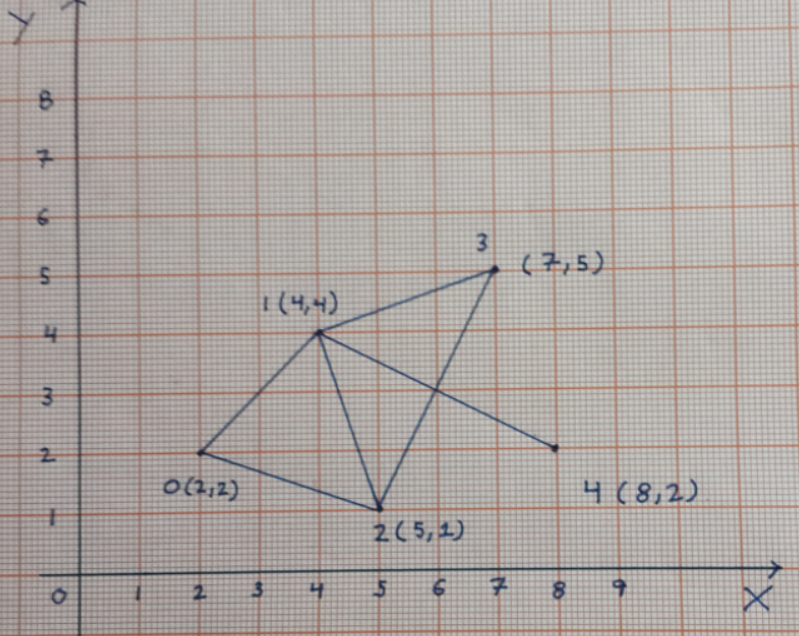
[['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1'], ['0', '0', '0', '0', '0', '1', '1', '1']]

Best: (-0.14907119849998599, ['0', '0', '0', '0', '0', '1', '1', '1'])

Final Solution : ['0', '0', '0', '0', '0', '1', '1', '1']

**Solution : 0 ---> 1 ---> 3**

2.



Number of Nodes : 5 (0,1,2,3,4)

Starting Node : 0

Ending Node : 4

Population after 1000 iterations(Population Size 40) :

[['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'], ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0']]

Best: (-0.1369757358544491, ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0'])

Final Solution : ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '1', '1', '0', '0']

**Solution : 0--->1--->4**

Limitations :

* We have considered the path length to be equal to the number of nodes present , while this gives the correct result , it is quite inefficient and takes longer to converge.
* Fixed chromosome length leads to inefficiency
* The Selection operator used is quite trivial and can be improved
* 2-Point Crossover instead of single point Crossover
* Mutation to ensure population diversity

Conclusion :

A simple genetic algorithm with fixed chromosome length is developed for a mobile robot to obtain a shortest path in 2-D static environment with obstacles. The code takes the co-ordinates and connectivity of the nodes as input and prints the shortest feasible path between the start and end nodes . Since GA is used to calculate the path the method is highly specific and adaptive .

References :

*1. Gihan NAGIB and W. GHARIEB "Path Planning for a mobile robot using genetic algorithm"*