7. GENETIC ALGORITHM BASED PATH PLANNING

7.1 INTRODUCTION

Genetic algorithms are search algorithms based on the process of natural selection and natural genetics [147]. They utilize the natural selection of the fittest individual among a group, for solving the problem in an optimal way. Optimization is achieved by the swapping of genetic material between parents. Genetic algorithms are based on the concept that individuals who are fit will live and the others will die. Genetic Algorithms can be used in all applications where optimization is needed.

7.1.1 The Steps involved in a Genetic algorithm

Step 1 : Start

Step 2 : Encoding of chromosome.

Solution for the problem is termed as the chromosome. Part of the solution is called the gene. Hence, the chromosome is also called a collection of genes.

The other name for the chromosome is ‘individual’. The solution is coded in a suitable way. The most preferred one is binary coding.

Step 3 : Assume the size of the population

Population is a group of individuals. There may be ‘n’ number of solutions to the problem. Out of that take k solutions (k<n) and fix them as the size of the population.

Step 4 : Design a Fitness Function
The fitness function can be designed in such a way, that it meets all the objectives of the problem to be optimized. It is problem specific.

Step 5: Initialize generation

Let \( G = 1 \) and consider all the individuals in the population.

Step 6: Estimate the fitness value for each Individual; the fitness value is calculated for each individual using the fitness function designed in step 3.

Step 7: Select the individuals based on fitness. Some of the individuals available in the population are selected, based on fitness by using some of the available selection methods.

Step 8: Obtain the next generation population by

a. Eliminating the two lowest fitness individuals and duplicating the two highest fitness Individuals selected, using the selection method, so that the population size is maintained.

b. Perform the following operations on this new population in the order

b.1 Cross Over – Exchanging part of the chromosome between parents

b.2 Mutation – Changing of the gene value by another value in a chromosome.

b.3 Inversion – Interchanging of the gene values in a chromosome.

Step 9: Increment the generation; i.e., Generation \( G = G + 1 \)
Step 10: Repeat steps 5, 6, 7 and 8 for the next generation population obtained in step 7 till the convergence condition is met or the generation reaches the maximum value.

Step 11: Decoding of chromosome.

Step 12: Stop

7.2 MODELING AND REPRESENTATION OF THE FOREST DOMAIN

In this work, the forest domain is divided into square grids or cells. The cell will contain a sensor placed at the center which will sense the fire if it occurs inside the cell. The sensor will know the location coordinates based on the first quadrant. The decomposition of the forest using 5 x 5 grids with coordinates based on the first quadrant is shown in Figure 7.2.1. It is assumed that the maximum size of the obstacle is not more than the size of the cell, both in width and length.

<table>
<thead>
<tr>
<th></th>
<th>4,0</th>
<th>4,1</th>
<th>4,2</th>
<th>4,3</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,0</td>
<td>3,1</td>
<td>3,2</td>
<td>3,3</td>
<td>3,4</td>
<td></td>
</tr>
<tr>
<td>2,0</td>
<td>2,1</td>
<td>2,2</td>
<td>2,3</td>
<td>2,4</td>
<td></td>
</tr>
<tr>
<td>1,0</td>
<td>1,1</td>
<td>1,2</td>
<td>1,3</td>
<td>1,4</td>
<td></td>
</tr>
<tr>
<td>0,0</td>
<td>0,1</td>
<td>0,2</td>
<td>0,3</td>
<td>0,4</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.2.1 Decomposition of the forest using 5 x 5 grids with coordinates based on the first quadrant

The actor is available at the bottom and its coordinates (0, 0) are the start point and the cell in which the fire occurs is assumed to be the end point. Once the fire occurs inside a cell, the corresponding sensor will send location coordinates to the actor. Thereby the actor knows both the start and the goal positions. Hence the path has to be found by the actor to reach that cell in order to extinguish the fire.
7.3 PATH PLANNING ALGORITHM

The flow diagram of path planning for the actor is shown in Figure 6.3.1.

Step 1 : Start

Step 2 : The path is encoded as a sequence of (via) points between the start and the goal positions; i.e., \((x_1, y_1) \ (x_2, y_2) \ldots \ (x(L-2),y(L-2))\) where \(L\) is the Length of the chromosome

Step 3 : Generate a number of chromosomes either by giving direct input solutions or solutions generated randomly, which form the population of chromosomes.

Step 4 : Let Generation \(G=1\)

Step 5 : a. Check for a chromosome in the population

    If (generated sequence of points for a chromosome is not in x-monotonic order) then

    Perform bubble sort for the sequence of points by considering the x values alone.

    Here we make use of the inversion operator only when two successive points in a chromosome are not in x-monotonic order.

    end

b. Repeat the step ‘a’ for all chromosomes in the population

Step 6 : Calculate the value of each individual based on the fitness function, which is a measure of the distance, and the presence or absence of an obstacle. While calculating the fitness, also check whether any via point contains an obstacle.

    If it contains an obstacle it is changed to negative points.
Step 7: Sort the individuals in the increasing order of fitness

Step 8: Select the first two individuals, reproduce them, and delete the last two individuals so that the population size of the chromosomes is maintained constant.
Figure 7.3.1 Flow diagram of path planning
Step 9: Perform the mutation operation only if the chromosome contains any negative point. The operation of mutation will change the negative point into positive point, and increment the y coordinate value in order to deviate from the obstacle.

Step 10: Perform the operation crossover

Step 11: Increment the generation; i.e., \( G = G + 1 \)

Step 12: if \( G < 25 \) then go to step5 else go to step13.

Step 13: Stop.

7.4 TERMINOLOGIES

7.4.1 Encoding of chromosome

In this work, the path is considered as a chromosome. A path can be considered as a sequence of points between the starting point and the ending point. The sequence of points between the start and the end points are called the via-points. Since there are many paths between the start and the end points, the number of via-points will vary for different paths. Hence, the chromosome can be variable in length or it can be fixed. In this work, fixed length chromosome is used. The reason for the fixed length chromosome is that the forest domain is decomposed into \( m \times n \) square grids/Cells. The length of the chromosome is usually taken as \( L = \) the number of cells in a row or the number of cells in a column, whichever is greater. Usually, the start and end points are not included in the encoding part of the chromosome. Also, we have used the integer coding of the chromosome rather than binary coding, which is considered to be more efficient. The via points are coded as genes. Therefore, the chromosome can be coded as shown below:

\[
(x_1, y_1), (x_2, y_2) \ldots (x_(L-2), y_(L-2))
\]  

(1)

where \( x_i, y_i \) – coordinates of the via point
7.4.2 **Fitness function**

In order to extinguish forest fires quickly, the following must be taken into consideration

a. The distance between the start and goal positions must be minimized.

b. The robot/Actor should avoid collision with obstacles

The fitness function used in this work is given by

\[ F = D + ON \]  

Where

\( D = \text{distance of path} \)

\( ON = \text{Number of obstacles along the path} \)

**7.4.3 Calculating the distance of the path for an individual**

Let us consider that \( d_1 \) is the distance between the start point and the first via point. \( d_2 \) is the distance between the first via point and the second via point. \( d_3 \) is the distance between the second via point and the third via point. \( d_n \) is the distance between the last via point and the goal point.

Therefore

\[ D = d_1 + d_2 + d_3 + \ldots + d_n \]  

Equation (3) is calculated for all Individuals.

where \( d_i \) is calculated as follows:

\[ d_i = \sqrt{\left(x_{i+1} - x_i\right)^2 + \left(y_{i+1} - y_i\right)^2} \]  

where
sqrt – is a function which will return the square root of a value taken as an argument

sqr – is a function which will return the square of a value taken as an argument

### 7.4.4 Calculating the number of obstacles

In this work, we have used global path planning; i.e., information about the environment is known a priori. The location of an obstacle in a particular cell is already known. This can be simulated in the program, by having a two-dimensional obstacle array, which contains a value of 1 if that particular coordinate of cell has an obstacle; otherwise, the value is 0.

Let us assume \((x_1, y_1) (x_2, y_2) \ldots (x_n, y_n)\) are the via points of a path. Then

\[
ON = \sum O [x_i, y_i] \quad \text{for } i = 1 \text{ to } n
\]

where \(O [x_i, y_i]\) is a two-dimensional array which will hold a value of 1 for the presence of the obstacle and a value of 0 for the absence of one. This ON is added to fitness value of that individual. Also when \((x_1, y_1)\) contains an obstacle it is changed to \((-x_1, -y_1)\).

### 7.4.5 Selection of parents for mating

Consider a population of \(K\) individuals with \(L\) as the Length of the individual. Calculate the fitness value for each individual. Based on their fitness, two individuals with high fitness are selected and reproduced. The two individuals with the lowest fitness values are deleted. So, the population size is maintained the same. The selection of individuals with high fitness is done by:

1. Ranking the individuals in the population in the increasing order
2. Selecting the first two individuals, reproducing them, and deleting the last two individuals.
7.4.6 Inversion operator

The Inversion operator is used to interchange the genes of the chromosome in the genetic algorithm. In this work the inversion operator is applied as the first step in each generation, in order to make sure that the genes in the individual are in the x-monotonic increasing order; i.e., if the robot is moving forward, and one via point changes its direction opposite to the goal or end point, then it will increase unnecessarily the length of the path. In order to avoid this, each and every chromosome will be checked as to whether the via points are in the x-monotonic order. The check is done by using the bubble sort technique to make sure that the sequence of the via points is in the x-monotonic increasing order. Here, the inversion operator is applied if the two successive genes are not in the x-monotonic order. The above process is repeated till all the individuals in the generation are in order.

7.4.7 Mutation operator

The mutation operator is used to change the value of a particular gene of the chromosome in the genetic algorithm. In this work the mutation operator is applied to check the individual to see if any gene of the chromosome contains an obstacle; i.e., the mutation operator will check for a negative via point. If there is any negative via point then it will make it into a positive point, and just add one value to the y-coordinate, in order to deviate from the via point which contains the obstacle.

7.4.8 Cross over operator

The cross over operator is used to exchange some of the genes between the two different chromosomes in the genetic algorithm. In this work the cross over operator is applied by randomly taking a position between the first via point and the last via point, both inclusive, and then from that position onwards exchange of genes takes place till the last via point between two chromosomes.
This position is calculated using a random number generation technique between 1 and the number of via points.

7.4.9 Finding an optimal solution

An optimal solution is found when the program terminates, either by achieving the convergence condition or when the number of generations reaches a maximum value. In this work, the convergence condition is when the individuals obtained in the current generation have the same fitness value as those in the previous generation. The number of generations can be calculated using the formula

\[ NG = L \times NC \]  

where

- \( NG \) – Number of generations,
- \( L \) – Length of the individual/chromosome,
- \( NC \) – Number of individuals/chromosomes.

In this work, \( NG = 25 \), and \( L=5 \), including both the start and end points, \( NC=5 \).

7.4.10 Remove operator

In this work, new operator called the Remove operator has been introduced which is applied only after finding an optimal solution. If there are two or more via points repeated in the optimal solution, only one via point is retained and the others are removed to eliminate duplication of the via points.

7.5 SIMULATION RESULTS

The path finding of the actor to extinguish the fire using the genetic algorithm is implemented in ‘C++’ Language. Algorithm is implemented on 5 x 5
grids because in [122] the 5 x 5 grid is used to explain the concepts. Further if we keep the size as small as possible we can explore the solution space thoroughly in order to validate the working of developed algorithm. The cells of the 5x5 grid are represented using the coordinates of the first quadrant. The presence of an obstacle in a particular cell is indicated by specifying a value 1 for the corresponding x and y coordinates of the cell in the obstacle array. The program will accept the start and end points from the user. The via points generation is carried out by (i) giving the chromosomes directly as the input which will form a solution (ii) randomly generating the input chromosomes. The algorithm is tested for 25 generations and placing an obstacle in the path. In both cases, paths are produced which are free from the obstacle, after the application of the algorithm. In general we can vary the grid size, length of the chromosome, and the number of generations. The chromosomes generated randomly produced paths of minimal length, when compared to chromosomes generated directly from solutions. The generated minimal path, in turn is used by the actor to extinguish the forest fire. The results are shown below and the graphs are drawn, using the MS-office Package.

7.5.1 Input chromosomes directly given, not generated randomly

Enter Start and Goal Points between (0, 0) to (4, 4)

Enter Start Point: 0 0
Enter Goal Point: 4 1
Enter obstacle positions 1 for presence 0 for absence:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Enter initial chromosomes:
0 1 1 1 3 1
1 1 2 1 3 2
1 0 3 0 3 1
Initial population of chromosomes including the start and goal positions

0, 0 0, 1 1, 1 3, 1 4, 1
0, 0 1, 1 2, 1 3, 2 4, 1
0, 0 1, 0 3, 0 3, 1 4, 1
0, 0 1, 0 2, 0 3, 0 4, 1
0, 0 1, 1 2, 1 3, 2 4, 1

Figure 7.5.1 Graph for the path which passes through an obstacle (direct input)

Chromosomes after 25 generations (with path length)
0, 0 1, 0 2, 0 3, 0 4, 1 4.414214
0, 0 1, 0 2, 0 3, 0 4, 1 4.414214
0, 0 1, 0 2, 0 3, 0 4, 1 4.414214
0, 0 1, 0 2, 0 3, 0 4, 1 4.414214
0, 0 1, 0 2, 0 3, 0 4, 1 4.414214

Minimal (optimal) path chromosome after applying the Remove operator

--------------------------------------------------------------------------------------
0, 0 1, 0 2, 0 3, 0 4, 1 4.414214

Horizontal axis - y coordinate
Vertical axis – x coordinate

In the above graph the start position is (0, 0) and the goal position is (4, 1)

Path1 consists of a sequence of intermediate points : (0, 1) (1, 1) (3, 1)
Path2 consists of a sequence of intermediate points : (1, 1) (2, 1) (3, 2)
Path3 consists of a sequence of intermediate points : (1, 0) (3, 0) (3, 1)
Path4 consists of a sequence of intermediate points : (1, 0) (2, 0) (3, 0)
Path5 consists of a sequence of intermediate points : (1, 1) (2, 1) (3, 2)
After executing the genetic algorithm for 25 generations, the optimal path is found out.

The optimal path consists of a sequence of intermediate points

(0,0) (1,0) (2,0) (3,0) (4,1)

7.5.2 **Input chromosomes generated randomly:**

Enter Start and Goal Points between (0,0) to (4,4)

Enter Start Point: 0 0

Enter Goal Point: 1 1

Enter obstacle positions 1 for presence 0 for absence:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

Initial chromosomes are generated randomly

Initial population of chromosomes including the start and goal positions

```
0, 0 1, 0 2, 0 1, 2 4, 1
0, 0 0, 0 3, 1 4, 3 4, 1
0, 0 1, 4 2, 0 2, 1 4, 1
0, 0 3, 2 4, 1 0, 0 4, 1
0, 0 4, 4 2, 1 4, 1 4, 1
```

Chromosomes after 25 generations (with path length)

```
0, 0 1, 0 2, 0 2, 0 4, 1   3.414214
0, 0 1, 0 2, 0 2, 0 4, 1   3.414214
```
Figure 7.5.3 Graph for the path which passes through obstacle
(random input)

Minimal (Optimal) path chromosome after applying the Remove operator
--------------------------------------------------------------------------------------
0, 0 1, 0 2, 0 4, 1 3.414214
0, 0 1, 0 2, 0 4, 1 3.414214
0, 0 1, 0 2, 0 4, 1 3.414214

Horizontal axis - y coordinate
Vertical axis – x coordinate

In the above graph the start position is (0, 0) and the goal position is (4, 1)
Path1 consists of a sequence of intermediate points
(0, 1) (1, 1) (3, 1)
Path2 consists of a sequence of intermediate points
(1, 1) (2, 1) (3, 2)
Path3 consists of a sequence of intermediate points
Path 4 consists of a sequence of intermediate points
(1, 0) (2, 0) (3, 0)

Path 5 consists of a sequence of intermediate points
(1, 1) (2, 1) (3, 2)

![Graph](image)

**Figure 7.5.4** Graph for the optimal path not passing through obstacle (random input)

After executing the genetic algorithm for 25 generations, the optimal path is found out.

The optimal path consists of a sequence of intermediate points
(0,0) (1,0) (2,0) (4,1)

### 7.6 Discussion

In this chapter, a path planning algorithm for a mobile actor to extinguish forest fire using genetic algorithm is presented. The fixed length chromosome has been used and the experiment has been carried out by
generating chromosomes (i) directly (as inputs) and (ii) randomly, and providing obstacles along the path. In both cases, a path is produced which is free from an obstacle after the application of the algorithm. The chromosomes generated randomly produced paths of minimal length, when compared to those generated directly from solutions. Domain specific genetic algorithm operators and a new operator REMOVE has been used only to find the optimal solution to eliminate the duplicate via points, all of which are helpful and produced the minimal path. In this work, simple selection process of sorting the individuals based on fitness and then selecting two individuals which maximize the fitness function, which will be considered for next generation, is used.