Genetic Algorithm Based Approach for Path Planning in a Static Environment

Reuben Georgi*

***Abstract*—The path planning problem involves finding an optimal and collision free path for an agent from its starting position to its destination. Genetic algorithms, which are commonly used for solving global optimization problems, are utilized in this project to solve the global path planning problem for a static environment. The results of taking a point based approach are presented for different environments along with the effect of varying certain operations and factors.**

***Index Terms*—Genetic Algorithm, Path planning, Global Optimization, Static Obstacle Avoidance**

I. Introduction and Motivation

A. Path Planning:

Path planning is one of the fundamental capabilities of many autonomous vehicles and is a crucial issue in the fields of robotics and automation. Path planning refers to finding a sequence of valid configurations or a collision free path through an environment with obstacles, from the specified starting position to a desired target or goal while satisfying certain optimization conditions.

The various methods used to solve the path planning problem can be distinguished according to the type of environment and the type of algorithm. The path planning problem can involve a dynamic or static environment, depending on whether any obstacles in the environment are moving or stationary. The planning problem could be considered as global or local, based on whether complete knowledge about the search environment is available. The problem considered in this project is that of a global path planning problem in a static environment.

B. Genetic Algorithms:

A genetic algorithm belongs to a class of algorithms known as evolutionary algorithms which are inspired by certain biological processes. John Holland introduced genetic algorithms in 1960 based on the concept of Darwin's theory of evolution and his student David E. Goldberg further extended the concept in 1989 [1].

Genetic algorithms can handle problems which have multiple local optima, a large number of parameters and cases where derivative information is not available. As a result, this technique is often used as an approach to solve global optimization problems.

Path planning using a genetic algorithm based approach has been studied in the past and there is plenty of existing work in this domain. In [2], the authors used a genetic algorithm to solve the planning problem and find the optimal path for a mobile robot to move in a static environment expressed by a map with nodes and links. However, their approach makes use of a table to keep track of points which are linked and every time the environment (or the arrangements of nodes) is changed, a revision of the table would be necessary. In another article [3], the authors use a similar approach with a grid but define the solutions in terms of directions rather than points.

*The author belongs to the School of Aeronautics and Astronautics, Purdue University

Another aspect considered in this project, is to see if varying certain operations and factors significantly affect the solution. The genetic algorithm can be implemented in a number of different ways and some of these approaches will be discussed in the next few sections. The aims of the project can be summarized as,

- Utilize a point-based approach to generate the shortest collision free path for agents
- Consider the effect of varying certain operations and factors on the solution

II. Problem Formulation and Main Results

A. Problem Formulation

In this project, the entire search environment is considered to be defined by a grid and this is a very common approach to this problem. In the grid, one point will be considered the start or initial point and another point will be the goal. The main objective is to find the shortest path between these points while avoiding collisions with any obstacles. This means that the obtained solution will be in the form of a sequence of points.

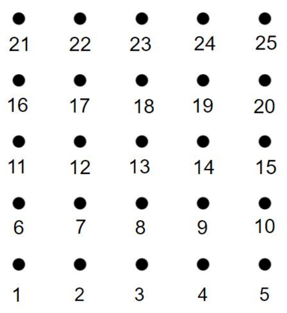

Fig. 1: Numbering of points in grid

Each point has its own x and y coordinate with point 1 having $x = 1$ and $y = 1$, point 2 representing $x = 2$ and $y = 2$ and so on. Any larger grids are defined in a similar fashion.

One of the benefits of using a grid is that it can define any kind of generic space or environment. Obstacles can also be easily defined and clearances can be incorporated by increasing the resolution of the grid.

The objective or the fitness function that is to be minimized is given by,

$$F = \sum_{i=1}^{t-1} d(P_i, P_{i+1}) + Penalties$$

Where t represents the numbers of points in a path and,

$$d(P_i, P_{i+1}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

This function calculates the total distance traveled by an agent on a certain path but there are also penalties for when the path is infeasible. The penalties are applied for paths which,

- Don't start or end at the positions which have been defined as the start and the goal
- Pass through an obstacle
- Move away from the goal
- Skip points/involve points which are not adjacent: For example, an agent can't start at point 1 and directly travel to point 16

B. Implementation of Genetic Algorithm

In the section, some of the important operations involving the genetic algorithm will be discussed.

1) Encoding: In a genetic algorithm each variable (point in this case) represents a 'gene' and these genes are linked together to form a 'chromosome', which represents a possible solution to the problem. Usually, in genetic algorithms, the variables are not used directly but are instead encoded and represented in binary as strings of 0s and 1s. This is the approach used in this project. However, other encodings are also possible such as real value encoding and tree encoding.

One of the consequences of using binary coding was that it limited the size of the grid to values equal to 2^n, where n is the number of bits used to describe each point. This number also directly affects the size of the chromosome used to represent the solution.

2) Initial population: Unlike some other optimization methods, an initial solution is not necessary for a genetic algorithm and the initial population of individuals/solutions is usually randomly generated. However, prior information about the problem can be utilized to improve the performance of the algorithm.

Thus, for the initial populations used in this project, the generated individuals had the required starting and final positions. The algorithm helped to find the optimal path between these points. For larger and more complicated grids, a suboptimal path had to be introduced into the initial population, in order to find a solution.

3) Selection: After generating the initial population, the next step involves the selection of solutions/individuals for the generation of a new population. The selection is based on the performance of the solution which is evaluated using the fitness/objective function. The aim is to select better solutions, in order to pass their characteristics on to the next generation.

The selection process can be implemented in a number of ways and the method utilized in this project is the tournament selection method. This involves two solutions being compared against each other and the better solution being selected.

4) Crossover: This operation enables the transfer of features of good solutions to the next generation. Solutions obtained from the selection process are taken two at a time and they are used to generate new solutions. As in the case of selection, the crossover operation can be done in a number of ways and there are three common types.

- Single Point Crossover: A point is picked randomly on the chromosome of both parents and the bits to the right are swapped, resulting in two new solutions with some traits of both the original solutions.

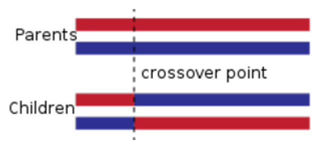

Fig. 2: Single Point Crossover [4]

- Double/Two-point crossover: This is similar to single point crossover with 2 points being randomly selected on the parent chromosomes instead of one.

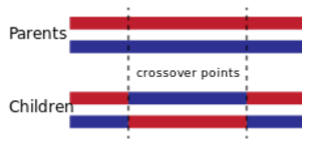

Fig. 3: Double/Two-point crossover [4]

- Uniform Crossover: There is an equal probability of a bit being inherited from either parent.

5) Mutation: This operation involves certain bits being flipped at random and helps to introduce new patterns into the population. It helps to improve the ability of the algorithm to find the global optimum. There are different types of mutation possible and bit string mutation was used in this project. A mutation in a solution may or may not make the solution better, but if the change is favorable it is likely to be passed on to the next generations. [5] suggests using the rate of mutation as $1/n$, where n is the number of variables, so that on an average, one variable gets mutated per solution. The main aim is to set a low mutation rate, as a high value can result in the search becoming random.

6) Termination Criteria: The termination criteria used to end the search is important as the genetic algorithm does not use any derivative information. In the project, the search is stopped if the best path or the best solution found does not change for a certain amount of generations.

C. Algorithms

In this section, the structure of the algorithms used for this project are presented. All of them were implemented in MATLAB.

Algorithm 1 Returns x and y coordinate of point

```
1: procedure CONVE ($a$,$np$,$nr$)
2: Generates $x$ and $y$ coordinates of points in grid with $np$ points and $nr$ rows using a structure array or struct.
3: Returns coordinates for input point $a$.
4: end procedure
```

Algorithm 2 Fitness Function

```
1: procedure DISTCALC ($P$, $s$, $g$, $obs$)
2: Use CONVE.m to find coordinates of all points in the grid
3: Use for loop to find total distance covered on path $P$
4: if first and last elements of $P$ do not match $s$ and $g$ then
5: Add problem dependent penalty to total distance
6: else if Distance between any two consecutive points in $P$ is greater than $\sqrt{2}$ then
7: Add problem dependent penalty
8: else if Distance to goal increases for successive points in $P$ then
9: Add problem dependent penalty
10: else if $P$ contains points belonging to $obs$ (obstacles) then
11: Add problem dependent penalty
12: end if
13: end procedure
```

Algorithm 3 Genetic Algorithm

```
1: procedure GENALG ($P_m$, $T_c$, $C_t$, $M_g$)
2: Generate initial population in binary with required starting and goal positions
3: for $i = 1$ to maximum number of generations, $M_g$ do
4:     Determine fitness value of all individuals in population using DISTCALC.m
5:     Store the best fitness function value obtained in the generation
6:     Check if number of generations with same value of best fitness is equal to $T_c$
7:     Apply tournament selection by comparing consecutive individuals in the generation
8:     Apply one of 3 crossover types based on value of $C_t$
9:     Apply mutation to current generation with a probability of $P_m$
10: end for
11: end procedure
```

D. Main Results

The results are presented in the form of 4 cases involving different problems or operations.

Case 1

- Grid Size/Number of Points: 16 (4×4)
- Starting Position: Point 1, Goal: Point 13
- Position of Obstacles: Points 5,6,9,10
- Termination Criteria: 25 consecutive generations with same best fitness function value
- Population Size: 200, Mutation rate: 0.0046

Population Size	**Path Length**	**Fitness func. evaluations**	**No. of generations**
100	3 units	4000	39
150	5.8284 units	6900	45
200	5.8284 units	7400	36
400	5.8284 units	14400	35

TABLE I: Case 1

Population Size	**Best Path**
100	[16,15,15,14,14,14,13]
150	[1,2,2,7,11,14,13]
200	[1,2,7,11,11,14,13]
400	[1,2,7,11,11,14,13]

TABLE II: Case 1 contd.

Case one considers a four by four grid and aims at exploring the effects of varying the population size. For a population size of 100, the genetic algorithm is unable to find any kind of path altogether. It is apparent that depending on the problem, the population size has to be large enough in order to find the solution. Also, there is some repetition of numbers in the best path but that's just a result of the fact that each solution consists of a set number of points and what is important to consider, is the order of the points which defines the path.

Case 2

- Grid Size/Number of Points: 16 (4×4)
- Starting Position: Point 1, Goal: Point 13
- Position of Obstacles: Points 5,6,9,10
- Termination Criteria: 25 consecutive generations with same best fitness function value
- Population Size: 200, Mutation rate: 0.0046

Crossover Type	**Uniform**	**Single**	**Double**
Best Path	[1,2,7,11,11,14,13]	[1,2,7,7,11,14,13]	[1,2,7,11,14,14,13]
Path Length	5.8284 units	5.8284 units	5.8284 units
Fitness func. evals.	7400	7200	6400
No. of generations	36	35	31

TABLE III: Case 2

Case two is very similar to case one in that it utilizes the same grid and has the same initial and final points. In this case, the aim was to see if there was any significant effect on the solution, by using different types of crossover. For this simple problem, with a population size of 200, all three types of crossover are able to find the solution. There is a small difference in the number of generations it took to find the solution and this, in turn affected the number of function evaluations.

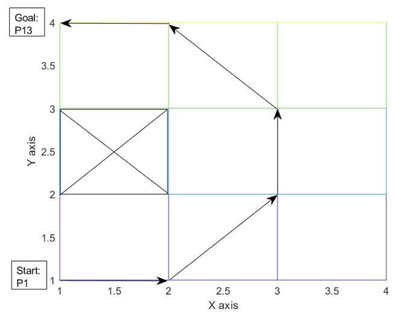

Fig. 4: Optimal Path for cases 1 and 2

This image presents the optimal path found by the algorithm for cases one and two. The obstacle is denoted by the box and the algorithm is able to find a collision free path from point 1 to point 13.

Case 3

- Grid Size/Number of Points: 32 (4×8)
- Starting Position: Point 2, Goal: Point 21
- Position of Obstacles: Points 13,14,17,18,23,24,28
- Termination Criteria: 20 consecutive generations with same best fitness function value
- Crossover type: Uniform, Mutation rate: 0.0032, Population Size: 300

Quantity	**Value**
Best Path	[2,6,11,15,19,22,22,21]
Path Length	6.8284 units
Number of fitness function evaluations	14,100
Number of generations	46

TABLE IV: Case 3

In case 3, a bigger 4 by 8 grid of 32 points is considered. The initial point is point 2 and the goal is to reach point 21. There are two obstacles in this case. The algorithm was able to find the optimal path and it took 46 generations to do so.

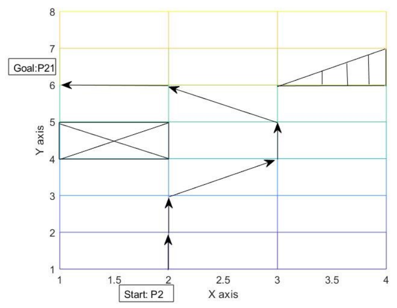

Fig. 5: Optimal Path for case 3

Case 4

- Grid Size/Number of Points: 64 (8×8)
- Starting Position: Point 2, Goal: Point 62
- Position of Obstacles: Points 11,18,19,29,30,38,37,36,46,45,44,49,50,42,41
- Termination Criteria: 20 consecutive generations with same best fitness function value
- Crossover type: Double Point, Mutation rate: 0.002, Population Size: 800

Quantity	**Value**
Best Path	[2,9,17,26,35,43,52,52,53,62]
Path Length	10.071 units
Number of fitness function evaluations	19,200
Number of generations	24

TABLE V: Case 3

In the final case, an 8 by 8 grid of 64 points was considered and the start and goal points were set to be 2 and 62 respectively. The environment contains 3 static obstacles.

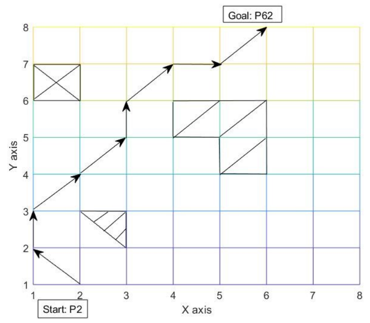

Fig. 6: Optimal Path for case 4

III. Your Ideas of Further Improvements

Here are some areas that have been identified for future work and for further improvement.

- Attempting a different approach by changing the operations or the formulation of the fitness function, in order to use this method for larger and more complex environments.
- Expanding the scope of the problem to consider dynamic environments, which may require an approach such as variable chromosome length or a real coded genetic algorithm.
- Using genetic algorithms for other applications. Genetic algorithms appear to be particularly appropriate for timetabling and scheduling problems, and many scheduling software packages are based on the technique.

IV. References

[1] Genetic algorithm, Wikipedia, the free encyclopedia , URL: https://en.wikipedia.org/wiki/Genetic_algorithm

[2] Nagib, Gihan & Gharieb, Wahied. (2004). Path planning for a mobile robot using genetic algorithms. 185-189. 10.1109/ICEEC.2004.1374415

[3] S. Choueiry, M. Owayjan, H. Diab and R. Achkar, "Mobile Robot Path Planning Using Genetic Algorithm in a Static Environment," 2019 Fourth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), Beirut, Lebanon, 2019, pp. 1-6

[4] Crossover (genetic algorithm), Wikipedia, URL: https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm)

[5] Deb, K.: 2001, Multi-Objective Optimization Using Evolutionary Algorithms, Chapt. 1, pp. 1–12. John Wiley & Sons, first edition.