Global Path Planning Optimization for Mobile Robot Navigation Using Genetic Algorithm

Shreya Reddy Bobbiti
Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
srbobbit@uwaterloo.ca

Abstract—Path planning is a famous NP-complete problem that is required for mobile robot navigation. The goal of this work is to study and find an effective solution to path planning problem in the case of mobile robot navigation, where the information about the environment is known by using a genetic algorithm. This paper explores the components of genetic algorithms such as mutation, crossover and selection operators by experimental implementation based on the study conducted on the past literature where path planning optimization was predominantly approached by using genetic algorithm. The model's design and performance has been discussed at length along with future work.

Keywords—path planning, global environment, genetic algorithm.

I. INTRODUCTION

Mobile robots are machines that can move around and perform specified tasks, they can be either autonomous or non-autonomous and can be used in a variety of environments such as land, ariel, and even underwater [1].

With the rapid advancements in the fields of IoT and Artificial Intelligence, mobile robot development has improved dramatically. At present, mobile robots find their usage in several industries such as, for transportation of goods in the manufacturing industry, for maintaining supply chain logistics in warehouses, exploration of an area for environmental monitoring, detection of landmines in the military, entertainment industry, space exploration, and even precision surgery.

Path planning is an optimization problem that is aimed at finding the optimal path along which the robot can traverse to reach its destination in an efficient manner, while also avoiding any collisions with the obstacles in the environment. And depending on the environments' information that is available to robot, path planning can be either of the two types: global or local [2].

Over the years several algorithms have been applied to model and solve this problem. Zhang et al, [2] have done an extensive review of all the methods that could be used to solve path planning. They found that of all the research, Genetic Algorithms are the most used method, this was determined by comparing the number of papers that use this methodology as retrieved from the database of Engineering Village.

Genetic algorithm is a stochastic artificial intelligence based evolutionary method that has been widely used to model path search. The algorithm starts with the genetic encoding of the path into binary bits and evolves based on the concept of natural selection. For global path planning, genetic algorithms were found to be useful as they have strong search capability and are useful to produce parallelism while being relatively simple to implement.

Another advantage of using genetic algorithms is that they are more likely to produce globally optimal solutions due to operators such as crossover and mutation, which makes them good at working with multi-modal optimization problems [6]. Since path planning of mobile robots requires multiple optimization criteria, as studied in past literature [3,4,5,7], such as the path length, path smoothness, path safety, also other criteria such as elevation (if the mobile robot is ariel instead of a ground robot), a genetic algorithm is a useful tool to model this problem.

II. LITERATURE REVIEW

Several works of research have been done in the past and the field continues to evolve. I have studied some of the works I found relevant to the path planning modelling.

Wahyu Rahmaniar et al [3], proposed a solution to the path planning problem using genetic algorithm. Here the researchers considered the mobile robots to be in an environment with multiple obstacles., where the information about the environment is known and initialized at first, then the user can set the starting and target points. They randomly generated and stored the positions of obstacles and added them to a population. They implemented a genetic algorithm by using local search with the path length and path smoothness as the two optimization criteria. They improved the path planning algorithm by using the Djikstra algorithm to optimize path length by considering the distance of robot from the obstacle to avoid collisions. It was observed that their proposed algorithm achieved better results and lower computation time as compared to traditional GA-based algorithms. However, the characteristics of sensors used in the mobile robot can be considered to further improve upon their proposed path planning algorithm.

Yanhui Li et al [4], proposed an improved method of genetic algorithm for mobile robot path planning in a static environment. In addition to the existing optimization criteria such as shortest path of the genetic algorithm the authors have added smoothness of the trajectory. Their approach had better convergence and smoother trajectory as compared to the basic genetic algorithm. The authors have used MATLAB simulation to verify their proposed algorithms performance

Yibo Li et al [5], proposed a genetic algorithm where A-star heuristic was used as the evaluation function for mobile robot path planning. This heuristic information that was added to the genetic algorithm resulted in acceleration of the convergence speed. They further considered a modification of the algorithm by adding insertion and deletion operators. They used a grid environment model to simulate the path planning algorithm in MATLAB. They compared the following results, the genetic algorithm with the A-star heuristic, the genetic algorithm with insertion and deletion operators and the traditional genetic algorithm. This comparison indicated that their proposed approach had improved the models' stability and efficiency by reducing the turning times and turning angles of the mobile robot.

Xiaolei Zhang et al [7], proposed a genetic path planning algorithm in a dynamic environment, where the obstacles are not stationary. They used the visible space concept and utilized matrix encoding to model the grid-based environment. Their proposed genetic algorithm uses a modified mutation operator that helps the mobile robot navigate in a dynamic environment while avoiding infeasible paths. Their proposed genetic algorithm is applicable in both static and dynamic environments. But they did not take into consideration the energy consumption, smoothness of path or the structure of the robot.

III. MOTIVATION OF RESEARCH

Of all the past study conducted some of the approaches where an improved performance was observed, when multi-optimization criteria (such as path length, path safety and path smoothness) were used together.

The plan for the remainder of this work is to study and explore the components of genetic algorithm that contribute to the successful implementation of robot path planning and observe how they affect these multiple optimization criteria when the environment is subjected to both static obstacles and when it is subjected to dynamic changes.

IV. EXPERIMENTAL MODELLING

The path planning of mobile robots constitutes of 3 major components: Environment Modeling, Selection of the Optimization Criteria, and Path Search Algorithm [2]. The selection of each of these components dictates how the algorithm designed would be of use in real-life scenarios.

A. Environment Modelling

As we previously saw, the application areas of mobile robots are very vast, ranging from several different land applications and extending to even ariel or underwater implementations. In these cases, not only the design of the robot such as its size, the sensors used and memory capacity of the robot but also the environment changes in accordance with the application, and it is hard to get a generalized model that is suitable for all the cases.

To reduce the complications in planning and for the purpose of simplifying the required number of computations, we can consider the environment to be a grid map on which the obstacles or the robot position can be marked with the help of coordinates in either two-dimensional space or three-dimensional space. Such a grid-based graph model would enable the entity to be able to traverse the area and reach a goal point specified by map coordinates.

The entity here (robot) is assumed to be a point in the map as it is not feasible to test the model on actual physical robot in this research work. Here an important consideration is the choice of obstacles that are scattered across the map, which are not considered as points but shapes with different dimensions represented using a group of map coordinates.

Further in global path planning problem the knowledge of the environment (map area) is presented to the robot. Obstacles positions are to be set prior to starting the navigation, and initially the obstacles are assumed to be static i.e., the only entity that would be moving across the map is the robot.

B. Optimization Criteria

There are usually several factors to be considered when designing an optimization criterion for path planning, As seen in the literature previously, the following are the basic criteria required for successful navigation by the robot.

1) Path Length: To find an optimal path there must be a set of coordinates that specify the starting location of the robot and another set of coordinates that specify the destination, it must travel to.

Since there are many ways, the robot could traverse to its destination it is important to find a path that has the minimum distance between start and end points, to get the best performance. This distance can be calculated by using different metrics such as Euclidean distance as shown in eq. (1).

$$D = \boxed{\sqrt{(x2 - x1)^2 + (y2 - y1)^2}} \qquad (1)$$

The path of the robot usually consists of many pairs of coordinates, specifying the free space waypoints between the initial and final points. And the total path length is the sum of all these individual distances between waypoints as given below in eq. (2) where 'n' represents total number of waypoints. And the fitness value is taken to be inverse of this length, signifying higher the path length, the lower it is desirable.

$$L = \boxed{\sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \qquad (2)$$

2) Path Safety: The main point in robot navigation is for the robot to be able to avoid the obstacles in the path that it must travel. Path safety is a criterion that ensures that there is a safe distance between the waypoint that the robot is choosing to travel to and the edge of an obstacle so that collisions do not occur.

This criterion is an important optimization condition because in real life the robots are not just points and have some physical size associated with it, this path safety distance would enable the algorithm to be extended towards real life robot navigation conditions. This is calculated by maximizing the minimum shortest distance between obstacles and waypoint across the map. This can also be translated as taking a negtaive of minimum to convert maximization to a minimization problem [8].

This minimum function can be translated to be a negative-exponential value so that as the distance varies and path comes closer to obstacle, the cost would be non-zero. If the minimum distance at which the robot can move around the obstacle is non-zero then it indicated there is a chance of colliding with the obstacle.

$$S = \boxed{- \mathrm{Min}(\mathrm{Min}\{\mathrm{Min\_D}(p_i, p_{i+1}, O_j)\})} \qquad (3)$$

3) Path Smoothness: If there are too many waypoints and the robot has to constantly change its direction it would reduce smoother travel and increase the time and energy required by the robot to travel to its destination. For this purpose, smoothness is considered an important criterion to be optimized, and it can be calculated by reducing the angle of rotation that the robot has to make [9].

The optimization criteria here would be the sum of all the angle of rotations of waypoint pairs [13], and it can be calculated as below shown in eq. (4).

$$\Phi = \boxed{\sum_{i=1}^{n-1} |\theta i|} \qquad (4)$$

Where,

$$\theta_i = \boxed{\frac{180}{\pi} \left(atan2\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) - atan2\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)\right)} \qquad (5)$$

This angle determines the change in the direction (rotation) of the line joining the waypoints.

Apart from these criteria, we make use of collision count value as discussed in Section V, in the genetic algorithm fitness function to get overall cost value.

C. Path Search Algorithm

Genetic algorithm being an evolutionary algorithm based over natural selection, allows for the evolution of algorithm, by utilizing selection, mutation, and crossover operators. It simply starts by generating random population of chromosomes and utilizing the fitness criteria to filter out the best set of chromosomes. Once the best chromosome is selected, it is modified (based on evolutionary operators) and allowed to move to next generation. The process is repeated until a stopping criterion is reached or the new generations shows no more changes.

As discussed previously, there exist several path search algorithms that have been implemented over the years. But the advantage of genetic algorithm over heuristic search approaches is that it does not require a lot of information to begin with, such as the exact location of the obstacles. This is very helpful in case when dynamic obstacles are present, as the algorithm would be capable of adjusting to these changes (by using evolutionary mutation and crossover operators), in newer generations. Furthermore, it's stochastic nature results in higher probability of finding a globally optimum solution.

V. PROPOSED METHODOLOGY

After deciding the environment, optimization criteria, and the path search algorithm, the next step is to model the path planning experimentally. I have used python programming language to build the genetic algorithm model. The flow chart of the experimental design is depicted the in Figure 1. And the major design components are explained as follows.

A. ***Environment:*** The environment is initially modelled as an empty 2D grid-based plane, over which the obstacles are generated and scattered across map based on the random initialization. The user interface waits for the user to enter the start and stop coordinates of the path to set the points, which will be read by the genetic algorithm and used to calculate the path for the number of generations inputted by the user.

B. ***Chromosome:*** The chromosome consists of different solutions to the optimization problem being solved [12]. In our case the chromosome represents a path which consist of a sequence of coordinate pairs, representing the path via starting, waypoints and end points on that map that robot travels along. The initial generation of chromosomes is obtained from a random uniform distribution encoded using decimal values.

C. ***Fitness function:*** The fitness function or the cost function is used to evaluate the set of chromosomes generated initially in the population and find the best chromosome such that optimization criteria is achieved.

A proposed addition to the cost function used in this algorithm is the number of collisions, the robot has with the obstacles while traversing the map. This addition to the fitness function heavily peanalizes the robot path that passes through the obstacles, resulting in higher cost value. The cost function is the sum of cost of all the three optimization criteria, i.e., inverse of total path length, path smoothness and path safety (minimum distance) and the collision count.

To further enchance the impact that these individual criteria have on the path, we can multiply the individual cost values by different integer values as shown in e.q (5), this can be modified according to the requiremnt of application. Here, the objective is to minimize this cost and converge to an optimal solution there by achiving zero collision value.

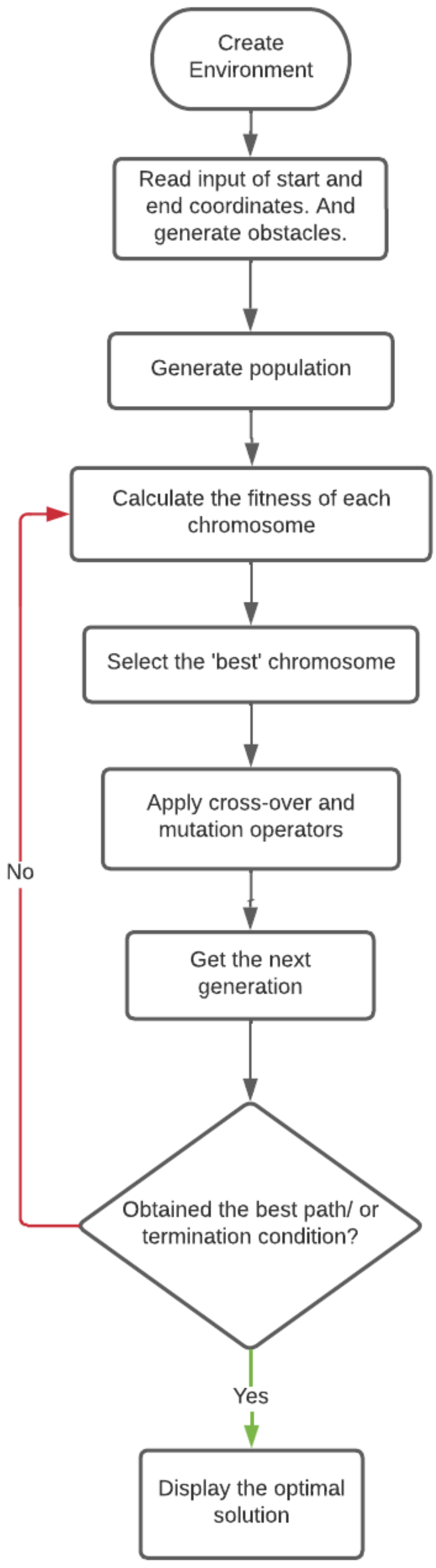

Create Environment
Read input of start and end coordinates. And generate obstacles.
Generate population
Calculate the fitness of each chromosome
Select the 'best' chromosome
Apply cross-over and mutation operators
Get the next generation
Obtained the best path/ or termination condition?
No
Yes
Display the optimal solution

Fig. 1. Gnetic Algorithm Path Planning Flowchart.

$$F = (N_c \times W_0) + (P_L \times W_1) + (P_{Sf} \times W_2) + (P_{sm} \times W_3) \quad (5)$$

Where, Nc = Number of Collisions,

P_L = Path Length,

P_{Sf} = Path Safety,

P_{sm} = Path Smoothness.

And W_0, W_1, W_2, W_3 are the weights of the respective optimization criterion. These integer weights can be varied according to the amount of penalization the application needs for each criterion, and best fit can be found by trial-and-error method.

D. Selection: The aim of genetic algorithm is to support the survival of chromosomes that are the fittest. In order to select the best chromosome, first the optimization functions are calculated, which are then used to calculate the fitness function value. And the chromosome having the lowest fitness cost value is selected into the population pool to produce the new best chromosome.

E. Mutation: The mutation operator produces genetic diversity in the population by arbitrarily flipping a gene value (in our case the coordinates of waypoint) resulting in generation of different chromosomal sequence [11]. Similar to how chromosomes were initialized at the beginning, this function randomly selects a gene and substitutes it with the a value from random uniform distribution.

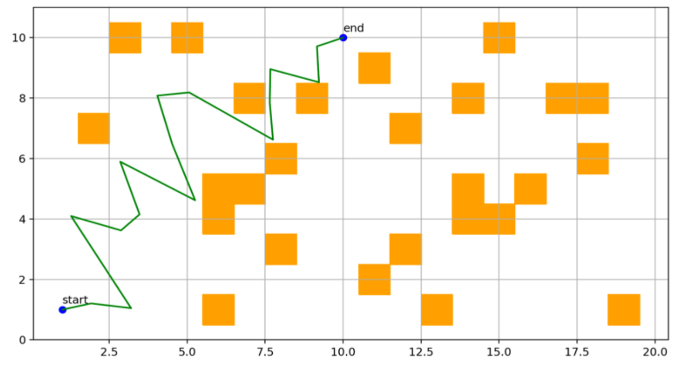
F. Crossover: In crossover operation the genetic information of two chromosomes (parents) is combined to generate a new offspring [10]. Depending on the number of crossovers it is of two types, single point crossover and two-point crossover. Here I simply took the mean coordinate value of both the parents to generate a new offspring point.

G. Optimal Solution*:* The most optimal path would be a minimal length smooth line joining the start and end points that the robot has to travel, while avoiding any obstacles by a safe distacnce. This ideal path would have minimum cost and zero collision value.

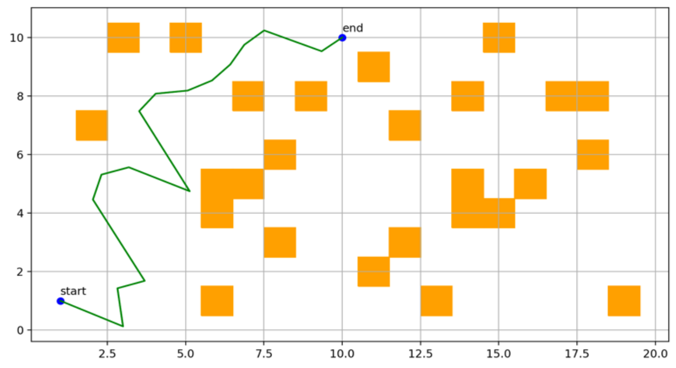
H. Convergence: A disadvantage of genetic algorithm is that the process of evolution is continuous and new chromosomes keeps on getting generated if there is no stopping criteria specified. To avoid this, the user is required to specify the number of iterations for which the genetic algorithm is ran, so as to reach convergence. Further, it is very difficult to achieve optimal solution because of the presence of a number of global paths the best we could do is achieve a near optimal solution within the given number of iterations.

VI. EXPERIMENTAL RESULTS

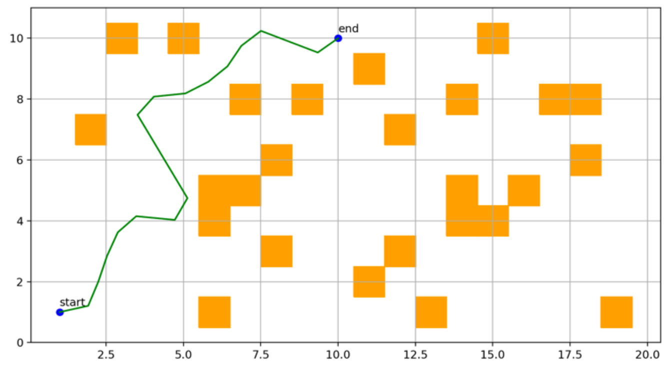
By following the above procedure, the simulation has been generated by making use of major Python libraries such as PyQt, Matplotlib and Shapely. The code has been attached in Appendix-1 for reference. For the purpose of reproduction of results, I have used a python seed to generate same obstacle map. But this can be modified to have different maps of obstacles as will be discussed below.



(a)



(b)



(c)

Fig. 2. Gnetic Algorithm Path Planning Simulation in static map for (a) 1st generation (b) after 30 generations (c) after 100 generations respectively.

At the start of the simulation, the GUI map remains empty as it awaits the instructions to be inputted by the user, such as to set the obstacles, set the start and end position coordinates of the robot path and to set the number of iterations for which the algorithm is to be run. Once this information has been set and the simulation starts, the calls to appropriate genetic algorithm functions are made such that a number of chromosomes are initially generated, out of which the best chromosome containing the coordinates of the path joining the starting and ending points is calculated such that it minimizes the cost function value.

Figure 2 shows the simulation results of the genetic algorithm for different generations. The orange blocks are the obstacles, blue dots signify the start and end points of robot as inputted by the user, and the line is the path generated using the genetic algorithm.

We can clearly see that the algorithm has been evolving to produce best fit path. And as the number of generations increases, the path also becomes smoother and maintains a safe distance from the obstacles.

After a certain number of iterations, there does not seem to be any significant improvement in the path, signifying that the algorithm has reached a convergence point, this was determined after viewing the results for generations [150, 200, 300, 500] as shown in Table 1. And hence it would be better to perform early stopping of the training instead of letting the algorithm evolve for very high number of generations, as increase in the mutations would only lead to oscillations in the algorithm.

TABLE I. FITNESS FUNCTION VALUES FOR DIFFERENT GENERATIONS IN STATIC MAP

Generation Number	**Total Cost**	**Path Length**	**Path Smooth-ness**	**Safe Distance**
1	129.25	26.70	2.43	0.99
10	126.77	25.88	2.43	0.99
30	116.37	22.59	2.50	0.92
60	98.65	18.34	2.11	0.90
100	86.02	17.45	1.25	0.92
150	85.97	17.44	1.25	0.92
200	85.61	17.32	1.25	0.92
300	84.63	16.99	1.25	0.92
500	79.39	16.26	1.00	0.92

The algorithm was run for different static obstacle maps (produced by changing the seed, so that the algorithms performance can be evaluated for different maps, some even complex ones) and the average time taken for the number of generations is obatined and tabulated in Table 2.

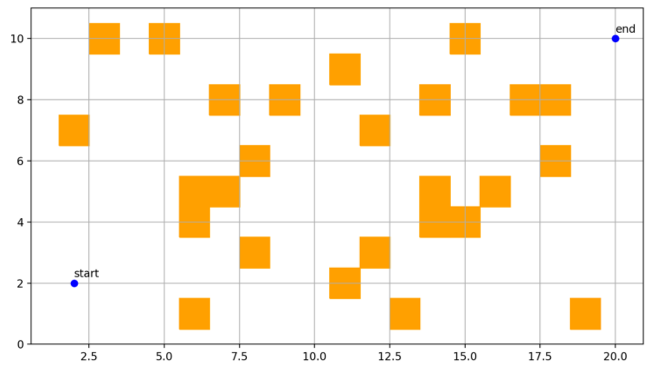
TABLE II. AVERAGE RUN-TIME FOR DIFFERENT GENERATIONS IN STATIC MAP

Generation Number	**Runtime in seconds**
1	0.2
10	3.3
20	6.6
30	9.9
40	13.1
50	16.4
100	32.2

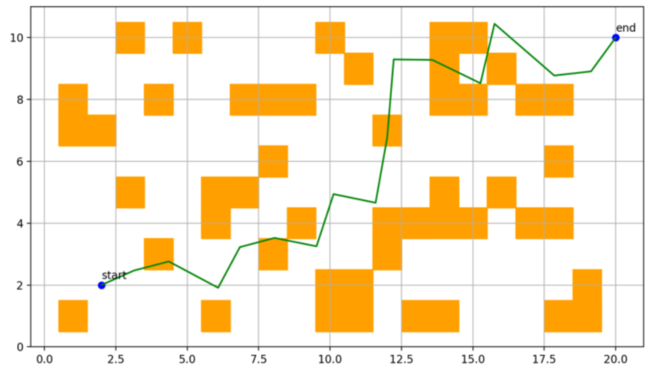
Next, the genetic algorithm has been subjected to changes in the obstacle map, mid-simulation by introducing a new function for dynamic shifting of the obstacles. The seed that was initially set to make sure same map is generated for every run is modified now to accommodate for these changes in the environment.

This dynamic function has been used after path generation, so that it causes the objects in the map to randomly shift their position. The number of times this change can occur can be controlled and for experimental purpose, the map has been subjected to 3 changes.

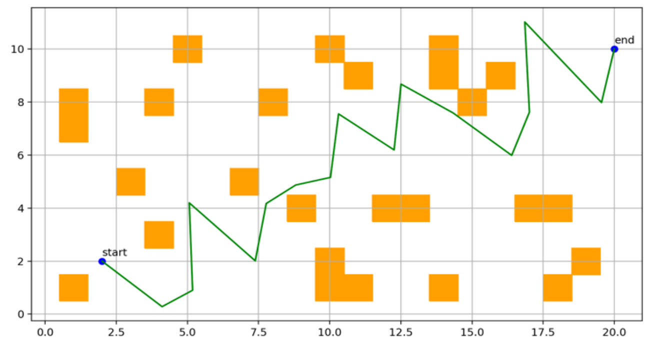
As shown in the Figure 3, the initial map changes as soon as the first generation finishes its execution, displaying the path, that has more number of collisions with the obstacles. This collisions occur due to unforeseen changes in the map encounterd by the genetic algorithm However, the future generations simply adapt to these changes in the environment by making use of evolutionary operators designed previously and will eventually generate a path that



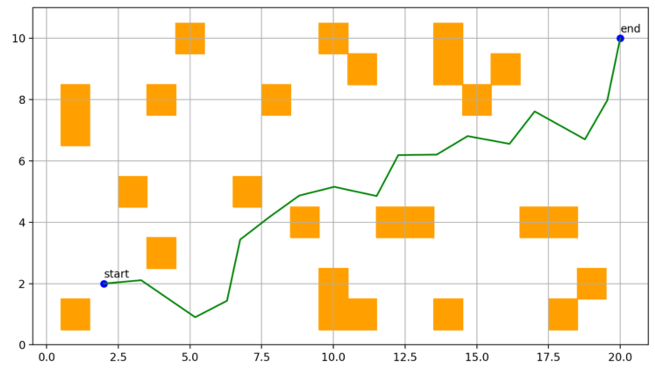
(a)



(b)



(c)



(d)

Fig. 3. Gnetic Algorithm Path Planning Simulation in dynamic map, depicting (a) initial map configuration (b) path after 1 generation (c) after 30 generations (d) after 200 generations.

does not collide with obstacles. The smoothness also improves after higher number of generations. After simulation the results for different groups of dynamic map settings, it was observed that the time taken for average generation runtime was similar to that of the static map time generated.

VII. DISSUSSION AND FUTURE WORK

Extensive literature has been studied before conducting this project work. Since there is already a substantial amount of research present on the use of genetic algorithm in case of path planning, it was interesting to explore how the best practices combined from these literatures could be put together into an algorithm. The evolutionary parameters of the proposed genetic algorithm have been modelled as per the best and simple experimental configurations of different papers explored previously, that gave rise to optimum results.

The proposed genetic path planning algorithm has successfully achieved the task of navigating through an environment full of obstacles and reach the goal point. This algorithm can be useful in real-life situations where the mobile robot knows the environment, i.e., the size of the area it is supposed to travel in, and the start and goal point it must reach. This information will help in tuning the models' parameters such as the population pool size of the genetic algorithm. For the experiments conducted above, the population size has been fixed to a value that has been decided after many trial and errors. The experiment can also be extended to larger map sizes, but for this work complexity was reduced so that faster and comparable results would be produced.

The introduction of dynamic shift of obstacles in the map has been applied to all the obstacles at once, but in real-life situations such as in an autonomous warehouse where the mobile robot worker would be required to move from point A to point B, the movement of obstacles would not be so dramatic as to shift all at once. In future, the individual coordinates of the obstacles can be pre-positioned (as required according to the actual layout of the warehouse or search space) and directed to shift at specified time duration and at dedicate speeds so that there would be an accurate depiction of the real-world path planning problem.

Moreover, the simulation environment itself could be extended to three-dimensional space (for example by using WEBOTS simulation tool) so physical space occupied by the robot and other environmental factors such as lag in the movement can be accounted for.

VIII. Conclusion

In this project, a working implementation of genetic path planning algorithm has been proposed based on best practices seen in previous works. The proposed algorithm is capable of working in an environment with both static obstacles and even adapts to when the obstacle map is subjected to dynamic shift. The key components of the genetic algorithm, such as the evolutionary selection, mutation, and crossover operators are explored and leveraged to suit the design needs of path planning. The experimental results achieved the goal of path planning where the robot's path from a starting point to desired ending point has been modelled in both non-moving and moving obstacle environments.

IX. References

[1] Brush, K. (2019, August 1). *What is a mobile robot? definition from whatis.com.* IoT Agenda. Retrieved October 7, 2022, from https://www.techtarget.com/iotagenda/definition/mobile-robot-mobile-robotics.

[2] Zhang, H.-y.; Lin, W.-m.; Chen, A.-x. Path Planning for the Mobile Robot: A Review. *Symmetry* 2018, *10*, 450. https://doi.org/10.3390/sym10100450.

[3] Rahmaniar, W., & Rakhmania, A. (2021). Mobile Robot Path Planning in a Trajectory with Multiple Obstacles Using Genetic Algorithms. *Journal of Robotics and Control (JRC),* *3*(1), 1-7. doi:https://doi.org/10.18196/jrc.v3i1.11024.

[4] Y. Li, Z. Huang and Y. Xie, "Path planning of mobile robot based on improved genetic algorithm," 2020 3rd International Conference on Electron Device and Mechanical Engineering (ICEDME), 2020, pp. 691-695, doi: 10.1109/ICEDME50972.2020.00163.

[5] Y. Li, D. Dong, and X. Guo, "Mobile Robot Path Planning based on Improved Genetic Algorithm With A-star Heuristic Method," 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), 2020, pp. 1306-1311, doi: 10.1109/ITAIC49862.2020.9338968.

[6] R, S. (2022, August 24). *Genetic algorithms and its use-cases in machine learning*. Analytics Vidhya. Retrieved October 7, 2022, from https://www.analyticsvidhya.com/blog/2021/06/genetic-algorithms-and-its-use-cases-in-machine-learning/#:~:text=By%20simulating%20the%20process%20of,problems%20faced%20by%20traditional%20algorithms.

[7] Zhang X, Zhao Y, Deng N, Guo K. Dynamic Path Planning Algorithm for a Mobile Robot Based on Visible Space, and an Improved Genetic Algorithm. ("Dynamic path planning over CG-Space of 10DOF Rover ... - Cambridge Core") International Journal of Advanced Robotic Systems. 2016;13(3). doi:10.5772/63484.

[8] Xue, Y., & Sun, J.-Q. (2018). Solving the Path Planning Problem in Mobile Robotics with the Multi-Objective Evolutionary Algorithm. *Applied Sciences*, *8*(9), 1425. https://doi.org/10.3390/app8091425

[9] Tuncer, A., & Yildirim, M. (2012). Dynamic path planning of mobile robots with improved genetic algorithm. *Comput. Electr. Eng., 38*, 1564-1572.

[10] Wikipedia contributors. (2022, May 3). Crossover (genetic algorithm). In *Wikipedia, The Free Encyclopedia*. Retrieved 16:41, October 17, 2022, from https://en.wikipedia.org/w/index.php?title=Crossover_(genetic_algorithm)&oldid=1085992909.

[11] Wikipedia contributors. (2022, June 2). Mutation (genetic algorithm). In *Wikipedia, The Free Encyclopedia*. Retrieved 16:21, October 17, 2022, from https://en.wikipedia.org/w/index.php?title=Mutation_(genetic_algorithm)&oldid=1091156249

[12] G. Nagib and W. Gharieb, "Path planning for a mobile robot using genetic algorithms," International Conference on Electrical, Electronic and Computer Engineering, 2004. ICEEC '04., 2004, pp. 185-189, doi: 10.1109/ICEEC.2004.1374415.

[13] Ou, J., Hong, S.H., Ziehl, P. *et al.* GPU-based Global Path Planning Using Genetic Algorithm with Near Corner Initialization. *J Intell Robot Syst* 104, 34 (2022). https://doi.org/10.1007/s10846-022-01576-6.