Collison-Free, Shortest-Path Planning for Mobile Robots in 2D Static Workspace using Genetic Algorithm

Technical report submitted in partial requirement of ENGR 635 by
Azmyin Md. Kamal
C00441440

1. Introduction:

A mobile robot is an intelligent vehicle that can perceive the workspace, acquire and interpret sensory data, determine its location within the workspace, and formulate a motion plan to satisfy a predefined objective. This objective may be moving with the least amount of energy, collision-free navigation, or coordinated movement with other agents [5, 8]. Classically mobile robots were restricted to manufacturing but in recent times, they have become increasingly ubiquitous in fields of medicine, entertainment, agriculture, mining, rescue, education, and military [7,8]. Hence, a mobile robot must be able to produce a feasible path from a starting point to a target position and reach that path safely and efficiently. Safely reaching the target position is one of the fundamental requirements of a mobile robot which makes choosing the navigation technique an important design decision. Figure 1 illustrates the basic building blocks of a mobile robot where path planning (red rectangle) is a key component in the overall process.

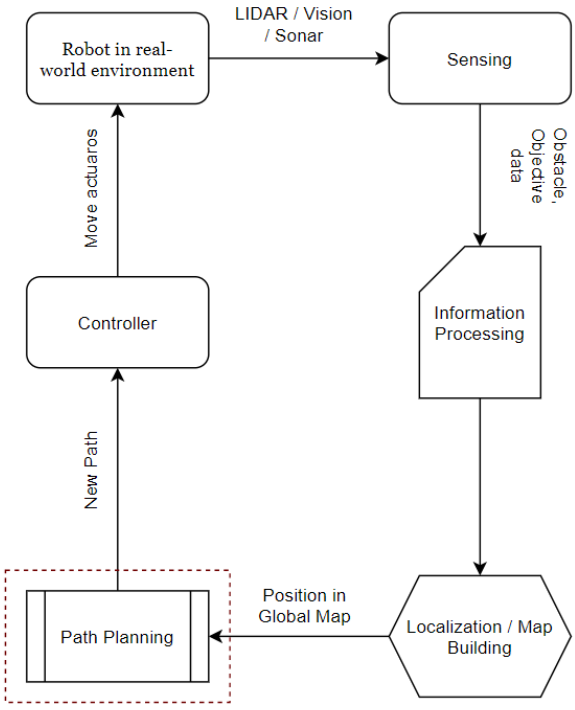

Figure 1: General architecture of a mobile robot

Patel et al. [7] and Nakhaeinia et al. [8] classified path-planning in mobile robots into two broad categories: Global path-planning (a.k.a Global navigation) and Local path-planning (a.k.a Local navigation). Lamini et al. define them as, “Global path planning consists in finding a reference path. It is determined before the agent begins its navigation by considering the environment as static. The second level (local planning) is based on the global path and data of the environment detected by the sensors to

form a dynamic occupancy grid" [5]. It is to be mentioned that for Global navigation, the complete information of the workspace is needed whereas Local navigation requires little to no prior information. However, Global path planning is an important step in overall navigation as pure Local path planning suffers from the issue of 'local trapping' which requires at least one known global path to solve.

Since the mid-1970s numerous techniques have been proposed and tested by researchers for solving both Global and Local Path Planning problems. Most of the proposed methods fall between two distinct categories viz. Classical and Reactive approach. Well-known classical techniques such as the A* algorithm, Road Map, and Artificial Decomposition are essentially grid-based search techniques guarantees to find a path or not solve at all. However, these techniques are computationally expensive (most utilizes some form of Exhaustive Search) and often fail in real-world dynamic conditions, thus making them less suitable for real-time implementations. Reactive approaches (a.k.a Heuristic Methods) are a class of methods, where the path planning is considered as an optimization problem. Notable methods in this class are the Genetic Algorithm, Firefly Algorithm, Particle Swarm Optimization, and Ant Colony Optimization [7,8].

Genetic Algorithm (GA) is a popular search-based optimization tool first introduced by Bremermann [9] and then applied to the field of computer science by Holland [10]. In the GA approach, a set of feasible and infeasible paths (characterized by a fixed or variable set of 'genes') constitutes the *population* in a *generation*. Each candidate in the *population* is then evaluated and assigned a fitness value depending upon the objective function. These individuals are selected as per their fitness value and allowed to pass their genes to a new generation using the *crossover*. A stochastic *mutation* operator maintains the diversity in the population and prevents premature convergence. Finally, the algorithm is terminated if the population has converged [8].

GAs have been used to solve both Global and Local path planning problems in literature [1, 8]. In this work, we solve the collision-free, shortest-path planning problem for two complex 2D workspaces using a Genetic Algorithm. Our primary assumptions are as follows

a. A grid-based 2D map where allowable vertices for each via point are known (Refer Figure 5 and 5)
b. If there are m obstacles and the obstacles can be considered as a point mass, the shortest path at most consists of (m+2) points
c. The first and last gene in all candidates remains unchanged as they represent the start and endpoints
d. At least one feasible path is found in each generation
e. The robot starting from via point 1, can only stay at the final point 15 multiple times but it is not allowed to stay on any other via points multiple times.

The primary contributions of this article are as follows

- A genetic algorithm capable of generalizing to n dimensions has been written from the ground up and tested with standard optimization functions

- Solved two complex 2D static map with good results
- Introduced two new heuristic rules to mitigate two major drawbacks of GA based path-planning

This technical report is organized as follows. Section 2 briefly illustrates relevant research into Global path-planning for a 2D static environment using Genetic Algorithms. Section 3 briefly touches upon a general Genetic Algorithm. Section 4 describes the two-step procedure taken to perform the experiments. Section 5 puts forward our experimental findings and Section 6 concludes the paper with indications for future work.

2. Background Study:

As introduced in Section 1, the Genetic Algorithm is a popular optimization technique that has been thoroughly explored by researchers, as they have proven to be superior to classical techniques in finding optimum collision-free paths in complex environments. The advantages of GA over classical methods are; ability to search a wide selection of feasible paths with more likelihood of finding the global optimum, strong optimization capability, and inherent parallelism [3,11]. The following is a short literature review of some prominent papers which investigated global optimum path-planning for a 2D static environment which is the premise of our experiment.

Robert Martin et al [13] used Probabilistic Roadmap (PRM) and Genetic Algorithm to compare their efficiency in solving the collision-free path planning problem for a simple and complex map. They showed that the GA algorithm produced smoother navigation and needed fewer turns to reach the final destination. However, the GA algorithm needed more time to find the best path in comparison to PRM.

Bakdi et al [12] demonstrated the efficacy of offline global path planning for a mobile robot based on a map dynamically generated using depth information obtained from two Xbox Kinetic cameras in an indoor workspace. They used RobuTER, a differentially driven robot that used onboard sensors and depth information from Kinetic cameras to successfully navigate through the complex workspace using the optimal path generated beforehand by the GA module. It is to be mentioned, the GA algorithm provided the initial path estimate and did not update during the robot's actual motion.

Lamini et al [5] proposed a new crossover operator for a Global optimal path planning problem in a 2D static workspace which took into account the variable chromosome length. crossover operator that uses parents with good fitness to increase the quality of progeny as well as parents with low fitness value to explore the research space and ensure diversity among the population. Experimental results showed their proposed operator and fitness function achieved convergence in fewer turns for three paths of increasing complexity.

Nagib et al [1] also investigated a Global path-planning problem for a 2D static environment where they considered 'via' points as 'genes' in a chromosome (each candidate path is a chromosome). Their primary objective was to find the shortest path without colliding with any obstacles in the workspace. The investigated path had 7 obstacles that corresponded to a chromosome of 9 genes thereby requiring 36bit representation. Experiments at 100% crossover and 0.05% showed good path-planning with a fitness

value of 55.71 units. However, the paper did not mention how many times the GA algorithm was able to find this result or how many feasible paths were introduced in Generation 1 to achieve this result. These shortcomings are critical in addressing the overall efficacy of any GA based Global path-planning algorithm.

Our work is based on the method presented in [1] but has been modified to add two additional heuristic rules to address the major shortcomings mentioned above.

3. Method: The Genetic Algorithm:

The following paragraph describes a standard genetic algorithm based on [2,5]

Genetic algorithms are naturally inspired algorithms which are based on the concepts of Darwinian evolution that involve an initialization method, fitness function to evaluate each chromosome, natural selection, crossover, and mutation operators [2]. The GAs begins by generating randomly an initial population that represents the possible solutions(chromosomes) to the problem to be optimized. Each chromosome is then evaluated by an adaptation function to determine the quality of every potential solution. After that, genetic operators are used to creating new progeny. The process starts with the stochastic (or deterministic) selection of the *parents* that will be subjected to reproduction according to their fitness values. Later crossover is applied to produce new *offspring* by recombining data from the two parents selected in the previous step. Mutation operator is then used to preserve the diversity of the population by changing the genetic structure of some individuals according to a mutation rate (usually a small value). This evolutionary cycle is repeated until the stopping criteria are satisfied, which can be the number of generations previously fixed otherwise, or the algorithm could be stopped when the population does not evolve quickly enough. The steps of a standard GAs [16] are detailed in Figure 3.

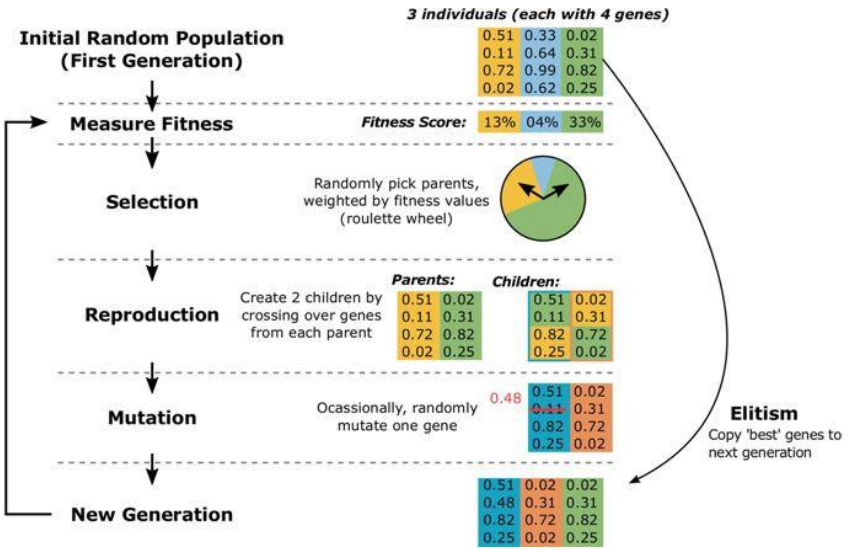

Figure 2: Basic Genetic Algorithm with Elitism [17]

4. Experimental Procedure:

Our experimental methodology is broken down into three subsections. In A, we verify the custom written Genetic Algorithm (GA) by solving a standard optimization problem. In B, the problem of solving the collision-free, shortest-path planning problem is formulated. In C, we discuss the modified GA used in this experiment

A. Verification of custom Genetic Algorithm:

We chose to test the Genetic Algorithm against De Jong's function 2 commonly called the Rosenbrock's valley (Figure 4). The global optimum lays inside a long, narrow, parabolic-shaped flat valley which can be located easily but convergence to the global optimum is very difficult for standard GAs. This occurs due to the standard GA's tendency to select fittest candidates with repetition that causes premature convergences to one of the local optima. Hence this problem is ideal to test the performance of our GA algorithm, which was written from the ground-up [14]. Table 1 details all the necessary variables, parameters, optimization function, and operators used in this test. Our GA algorithm was shown to be able to solve this problem but did suffer from a premature convergence problem. More details are given in Section 5 A

Table 1

Element	Description
Features	• Single-point cross over • Bitflip mutation operator • True Elitism • Roulette Wheel Selection • Supported gene encoding = 8bit, 16bit • Number of design variable supported = *n*
Design variable	$-2.048 \leq \boldsymbol{x_i} \leq 2.048$ where $\boldsymbol{x_i} = (x_1, x_2)$ *are the two design variables*
Parameters	U = upper bound, L = Lower bound, N = Number of candidates, num_iter = number of iterations to perform, (target_x1, target_x2) = coordinates of known global optima; Nx = Number of GA executions for explorative analysis
Objective Function	$f(x_1, x_2) = 4000 - \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2) + (1 - x_i)^2$ Here the objective is to MAXIMIZE the objective function. The integer 4000 converts the original Rosenbrock function from a minimization function into a maximization function
Global minima	$f(x_1, x_2) = 0$ for $\mathbf{x_i} = 1, i = 1, ..., n$
Crossover operator	A value between 1 to (n_bits – 1) i.e. for 16bits, a value between 1~15
Mutation operator	Flip bit between positions 1 - 16

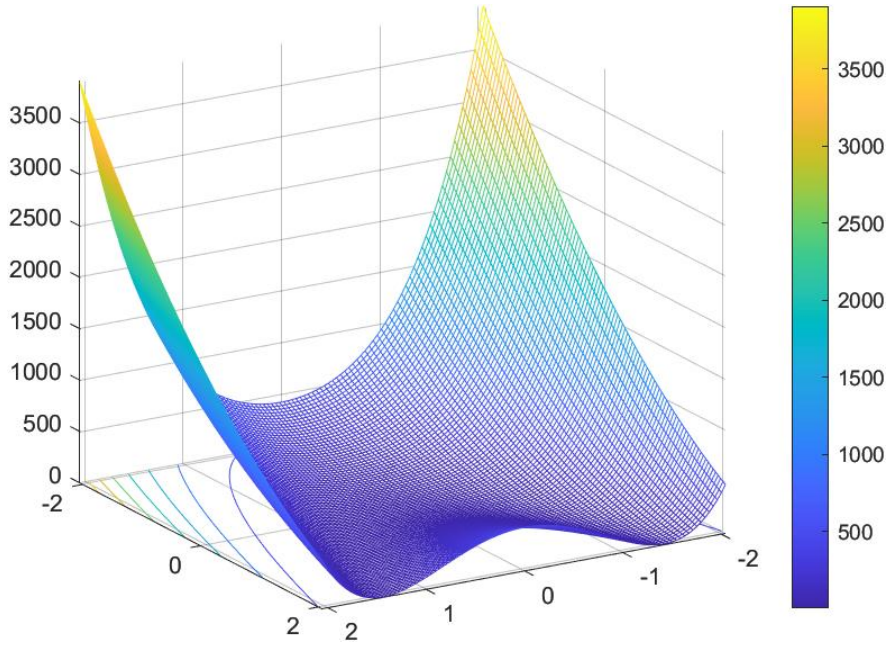

Figure 3 Rosenbrock Valley

B. Problem formulation:

The problem of global path-planning in a 2D static environment can be formulated as a maximization problem where the goal is to plan a collision-free shortest-path that would require the least number of via points to reach the final destination. The map for the two static complex environments is shown in Figures 5 and 6.

As shown in the figures mentioned above, the via points are marked from 1 through 15. The starting location is 1 and the ending location is 15. A string of points starting from 1 followed by intermediate via points and ending in 15 represents a path. For example, [1-2-3-4-5-6-7-9-15] is a path (Refer Figure 5). Each via point has a corresponding (x,y) coordinate and we choose a 4bit binary number to represent each via point. We assume the path joining two via points is always a straight line to conform to the Euclidean distance formula. Table 2 shows the (x,y) coordinate along with the binary encoding information for Workspace 1 (Figure 5).

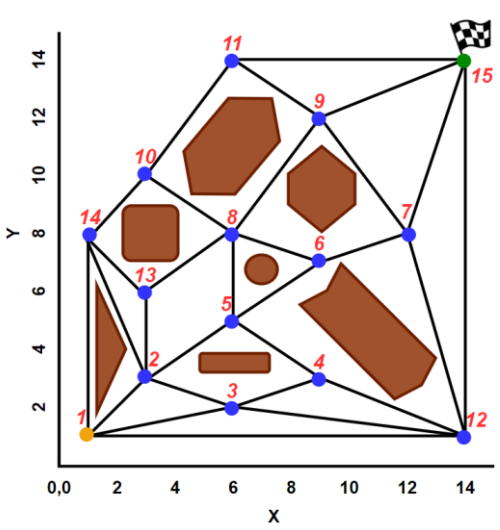

Figure 4: Workspace 1

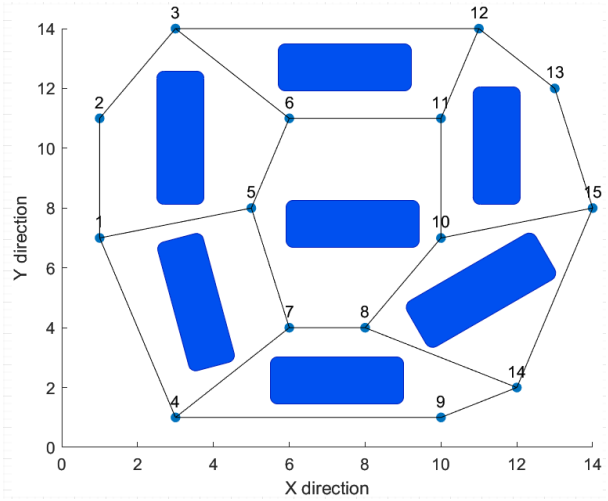

Figure 5: Workspace 2

Table 2: Binary and 2 axes coordinate map			
Via point	Binary representation	X coordinate	Y coordinate
1	0001	1	1
2	0010	3	3
3	0011	6	2
4	0100	9	3
5	0101	6	5
6	0110	9	7
7	0111	12	8
8	1000	6	8
9	1001	9	12
10	1010	3	10
11	1011	6	14
12	1100	14	1
13	1101	3	6
14	1110	1	8
15	1111	14	14

The starting location and the ending location in a string mustn't change during the genetic exchange process. We define the C-space as the robot's all possible paths that it can traverse in a given map. Our objective now can be defined as finding the optimal path in the C-space which has the highest fitness score. Paths that intersect with obstacles are termed as 'Infeasible' paths and will have a predefined low score. Having a value of 0 for an infeasible path introduces the 'divide-by-zero error' i.e. *inf* values.

C. Modified GA algorithm for Path Planning

One of the fundamental design decisions that must be considered when implementing GAs for a path-planning problem is whether to use binary encoding or perform operation directly on the integer string. According to [1,7], the binary encoding schema involves encoding the path from an integer string into a long binary string, perform operations (roulette wheel, crossover, and mutation) and decode the resultant strings back into integer string (refer Figure 7). We have adopted this approach as it naturally conforms to the multitude of crossover and mutation techniques discussed in literature. A binary representation of a *path* is called a *chromosome* [6].

Apart from which encoding schema to use, the length of the chromosome is also important as implementation GA varies depending on the length which can be fixed or variable [6-8]. For 4bit representation, a total of 16 via points are possible. The maximum chromosome length can then be *(16 * 4) = 64bits*. For fixed chromosome length implementation, there is no consensus on how to choose the maximum length of the shortest path. For instance, Nagib et al proposed chromosome length for the shortest path should be $(m+2) * n\_bits$, where m is the number of static obstacles and n_bits is the number of bits to represent the via points [1]. We have adopted this approach in this experiment.

Another key design choice is the size of the population in each generation. Classically this was a fixed value with the first generation being randomly generated [5]. We have kept the number of populations fixed but have opted to insert some known paths during the initial population generation. Without the inclusion of any valid path, the GA algorithm fails to find a solution or finds a very poor path for a given number of generations (usually 20 or 50 according to literature). More discussion on this issue is provided in Section 5 B.

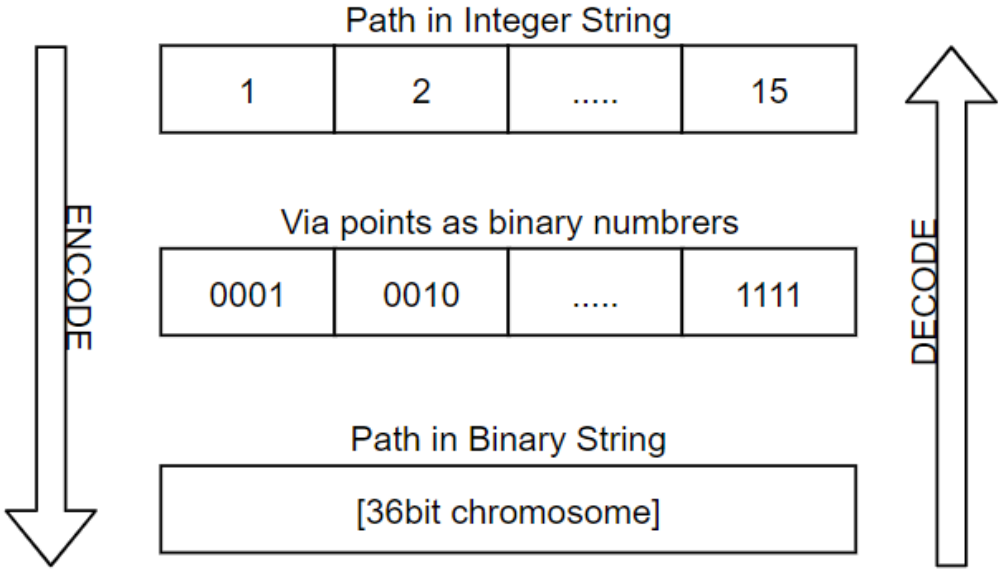

Figure 6: Encoding-Decoding schema in GA based path-planning problems

The parent selection operator used is the classical *Roulette Wheel Selection* operator that prefers to choose candidates with greater fitness. This is method is also *selection with replacement* from combinatorics, meaning a fit individual may be chosen multiple times from the mating pool and may mate with itself (i.e. no new offspring). For the crossover operator, we have chosen the *Two-point Crossover* technique which randomly picks two binary numbers from bit position 5 to 32. After crossover, a Bitflip mutation operator is chosen which randomly flips a bit between 5 to 32.

For sake of convenience, we present Table 3 which contains the details of the parameters used in our modified GA. Note that, the two new heuristic rules dubbed 'check_minima_lock' and 'inject_global_best' will be discussed in Section 5 B for better clarity.

Table 3

Element	Description
Features	• Two point crossover • Bitflip mutation • Roulette Wheel Selection • Number of bits to represent via points = 4bits • Two new heuristic rules
Design variable	A sequential integer string viz [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 to perform, known_solution = 4, Nx = Number of GA executions for explorative analysis
Objective Function(s)	$$f(x) = \begin{cases} \frac{1}{\sum_{i=1}^{m+1} d(P_i, P_{i+1})} : Feasible\ path \\ 0.001 : Infeasible\ Path \end{cases}$$ $$Fitness\ F(x) = 1000 * f(x)$$ $$d(P_i, P_{i+1}) = \sqrt{(X_{i+1} - X_1)^2 + (Y_{i+1} - Y_1)^2}$$ Here $d(P_i, P_{i+1})$ is the Euclidean distance between two via points in the path string. We are trying to MAXIMIZE *F(X)*
Global Best path (Workspace 1)	[1 – 2 – 5 – 6 – 7 – 15 – 15 – 15– 15] → F(X) = 50.72
Global Best path (Workspace 2)	[1 – 5 – 7 – 8 – 10 – 15 – 15 – 15– 15] → F(X) = 55.71
Crossover operator	A value between 5 ~ 32
Mutation operator	A bit between bit positions 5~ 32

5. Results and Discussions:

All three experiments were done in MATLAB 2019 Rev B in a computer containing Core i7-9750H CPU running at 2.59 GHz. It is to be noted that, the code for this experiment was written all from scratch with the use of some standard MATLAB function. Section 5 A discusses our results that verified our GA and Section 5 B will discuss the results obtained in

A. Verification of Genetic Algorithm with 2D Rosenbrock Function

To verify our custom GA algorithm is working correctly, we solved the problem of finding optimal minima of 2D Rosenbrock function with parameters shown in Table 4. Figures 8 - 11 illustrates

sample output from this stage of the experiment. Note that, the version of the code used to perform these experiments were artificially locked to solve 2 variable design problem but can solve n variable design problem, given the objective function is defined in a standard form.

Table 4	
Name	Value
N	10
iter_no	[5,10,15,25]
Crossover probability	90%
Mutation probability	0.001 ~ 0.006%
bit_count	8
U	2.048
L	-2.048
global (x1,x2)	(1,1)

DataTable = 10×6 table

	candidate_index	x1	x2	fit	fractional_fit	cumulative_prob
1	1	1.1645	0.2811	3.8844e+03	0.0982	0.0982
2	2	1.1645	0.9236	3.9813e+03	0.1007	0.1989
3	3	1.1645	1.3734	3.9999e+03	0.1012	0.3001
4	4	-1.1806	1.1645	3.9900e+03	0.1009	0.4010
5	5	-1.1806	1.2288	3.9925e+03	0.1010	0.5020
6	6	1.1645	1.3091	3.9998e+03	0.1012	0.6031
7	7	0.1365	1.4537	3.7933e+03	0.0959	0.6991
8	8	-1.0521	1.9356	3.9271e+03	0.0993	0.7984
9	9	-1.3091	1.2288	3.9711e+03	0.1004	0.8988
10	10	1.1645	1.3091	3.9998e+03	0.1012	1.0000

ResultTable = 1×2 table

	fitness_sum	fitness_average
1	3.9539e+04	3.9539e+03

Figure 7: Tabular data output during the verification stage

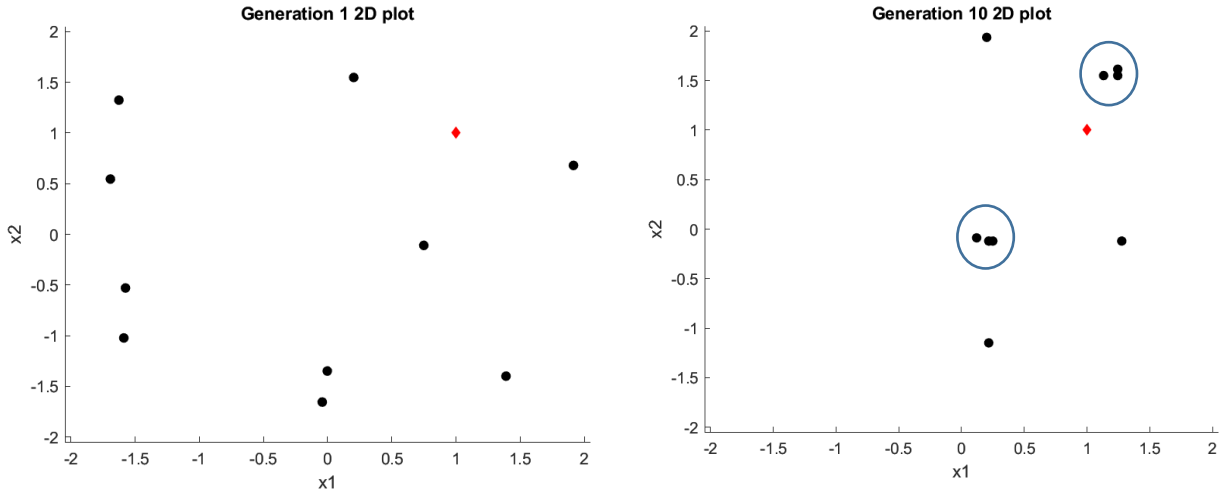

Figure 8 (a,b) – Top-down view of the location of candidates in solution space. The red diamond is the known global optimum. Issue of minima lock marked in blue circles

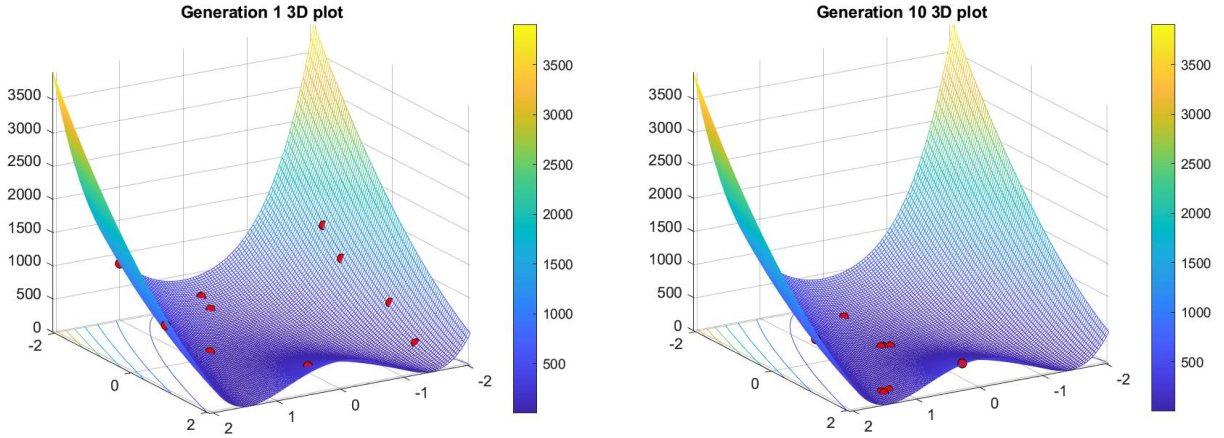

Figure 9 (a,b) – 3D view of candidate locations on the Rosenbrock surface when itern_no = 10.

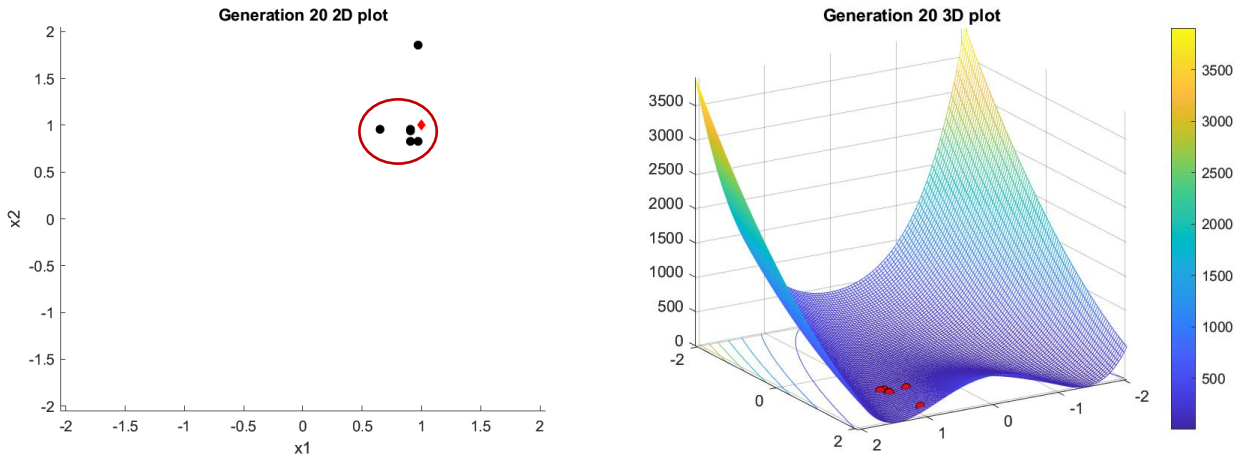

Figure 10 (a,b) – Results after iter_no = 20, near perfect result

Table 5 Results from 4 trial run				
Exp No.	iter_no	N	Best (x1,x2)	Avg. Fitness at final generation [0 ~ 4000]
1	5	10	1.1645, 1.3734	3953.9
2	10	10	1.2449, 1.5501	3917.3
3	15	10	-0.2881, -0.0241	3963.9
4	20	10	0.9075, 0.8272	4000

Table 5 clearly shows our custom Genetic Algorithm is indeed working correctly and has converged across all 4 experiments with very good results. It is also evident that increasing the number of generations improves the overall fitness, with the best candidates nearly coinciding with the global optima (Figure 10). However, from Figure 10, it is also clear that the Genetic Algorithm is plagued with the issue of premature convergence which may be solved by setting an aggressive mutation operator or introducing an additional function guard.

B. Collison-Free Shortest-Path Planning

Table 6 shows the parameters used for solving the optimal path-planning problem in Workspace 1.

Table 6: Parameters for Workspace 1	
Name	Value
N	10
iter_no	50
Crossover probability	90%
Mutation probability	0.001 ~ 0.006%
bit_count	4
s_loc	1
e_loc	-2.048

We introduced two heuristic rules ‘check_minima_lock’ and ‘inject_global_best’. These rules were made to exploit both the premature convergence issue and relative faster convergence caused by the Roulette Wheel Selection operator.

From Figure 9, it is apparent that Roulette Wheel’s tendency to repeat higher fitness candidates leads to premature convergence. This issue is very problematic for a path-planning problem where the workspace is characterized with integer via points. Due to premature convergence, in successive iterations, most of the candidates becomes the same path thus losing the capacity to find a better path very early into the iterative search. *check_minima_lock* counts the number of paths that have the same fitness value and if 80% of all paths are the same, it resets the whole generation.

inject_global_best solves the apparent loss of information caused by check_minima_lock which effectively erases any best path found till that generation. *inject_global_best* compares the current best against the current *global best* path. If the *global_best* has better fitness, this function overrides the first candidate of the current generation with the *global_best* path. If the current generation’s elite

has better fitness, the *global_best* path is updated with the current generation's best. Thus, even if the whole candidate matrix is reset, the *global_best* path is still preserved. We have observed better convergence with the use of these two heuristic rules. Figure 11 (a-d) illustrates the four common paths found in our experiment with path 11 (d) being the best.

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

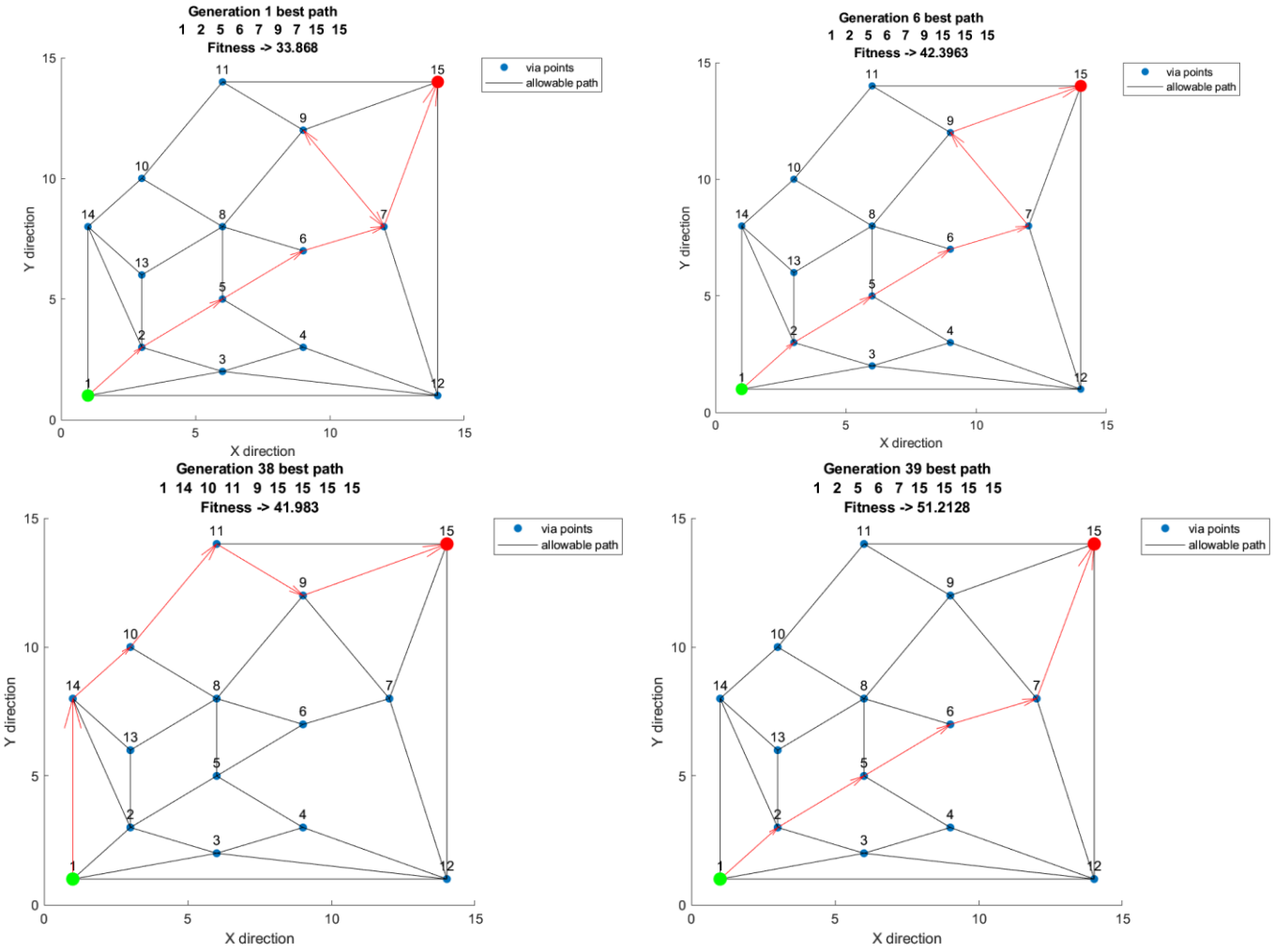

Figure 11 Results from our simulation. Figure 11a is a feasible path common in all experiments. Figure 11b is the most common optima found. Figure 11c is an alternative optimum path. Figure 11d is the global best for Workspace 1

Table 7 shows the result of the optimal path-planning algorithm with the heuristic rules turned on. Each generation has 10 candidate members with Generation 1 having three known *feasible* paths.

Table 7: With heuristic rules, 4 known paths			
No #	No. of Generations	Best Path @ Gen 50	Found global best?
1	50	Fig 11 d	Yes
2	50	Fig 11 b	No
3	50	Fig 11 b	No
4	50	Fig 11 b	No
5	50	Fig 11 b	No
6	50	Fig 11 b	No
7	50	Fig 11 b	No

It is evident from Table 7 that the use of two heuristic rules did help the GA algorithm to always converge to the 2nd best solution in workspace 1. Possible reasons for this premature convergence could be attributed to the Elitism strategy employed along with the *inject_global_best* heuristic rule. The *inject_global_best* did make sure we did not lose the global best-known path but it had also interfered with the method by which the Elitism operator would operate. Additionally, out of the 7 trial runs, the GA was able to find the global optima only once. The cause for this inability to find the global best multiple times is unknown at this time.

6. Conclusion with Future Work:

In this technical article, we modified a standard Genetic Algorithm to solve a global path-planning problem for a robot traversing through a 2D static environment. The primary objective was to find the most optimum path that would guarantee collision avoidance while taking the least number of via points to reach the destination. At first, a GA algorithm written from ground up was verified against the De Jong's function 2 where it should good results in finding the global optima within 20 ~ 25 generations. We then implemented a modified version of the GA to solve the proposed problem. Results indicated that the algorithm was able to find good paths but they were not the global optimum path. Future studies may investigate adding SUS parent selection operator, Tournament Selection operator [13], and change the starting and ending points to ensure the modified GA can still find the best possible path [1]. In this regard, a vertex search algorithm may be implemented to automatically generate a few feasible paths for the GA to function.

7. References:

1. Nagib, Gihan, and W. Gharieb. "Path planning for a mobile robot using genetic algorithms." IEEE Proceedings of Robotics (2004): 185-189.
2. Dr. T. L. Chambers, Lecture Slides on Genetic Algorithm. Available at: https://moodle.louisiana.edu/course/view.php?id=60503

3. Li, Qing, et al. "An improved genetic algorithm of optimum path planning for mobile robots." Sixth International Conference on Intelligent Systems Design and Applications. Vol. 2. IEEE, 2006.
4. Goldberg, (1989), “Genetic Algorithms in Search, Optimization and Machine Learning”
5. Lamini, Chaymaa, Said Benhlima, and Ali Elbekri. "Genetic algorithm based approach for autonomous mobile robot path planning." Procedia Computer Science 127 (2018): 180-189.
6. Tu, Jianping, and Simon X. Yang. "Genetic algorithm based path planning for a mobile robot." 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422). Vol. 1. IEEE, 2003.
7. Patle, B. K., et al. "A review: On path planning strategies for navigation of mobile robot." Defence Technology 15.4 (2019): 582-606.
8. Nakhaeinia, Danial, et al. "A review of control architectures for autonomous navigation of mobile robots." International Journal of Physical Sciences 6.2 (2011): 169-174.
9. Bremermann HJ. “The evolution of intelligence. The Nervous system as a model of its environment,” Technical Report no.1, contract no. 477(17). Washington, Seattle: Dept. Mathematics, Univ.; July 1958.
10. J. H. Holland, “Adaptation in natural and artificial systems. Ann Aebor,” MI: University of Michigan Press.
11. Santiago et al, "Path planning for mobile robots using genetic algorithm and probabilistic roadmap," 2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), Manila, 2017, pp. 1-5, doi: 10.1109/HNICEM.2017.8269498.
12. Bakdi, Azzeddine, et al. "Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control." Robotics and Autonomous Systems 89 (2017): 95-109.
13. Moghadampour, Ghodrat. "Outperforming Mutation Operator with Random Building Block Operator in Genetic Algorithms." International Conference on Enterprise Information Systems. Springer, Berlin, Heidelberg, 2011.
14. Moré, Jorge J., Burton S. Garbow, and Kenneth E. Hillstrom. "Testing unconstrained optimization software." ACM Transactions on Mathematical Software (TOMS) 7.1 (1981): 17-41.
15. Lee, Jaesung, and Dae-Won Kim. "An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph." Information Sciences 332 (2016): 1-18.
16. 17. A. Bal (2019), Demystifying Genetic Algorithms to enhance Neural Networks. URL - https://medium.com/intel-student-ambassadors/demystifying-genetic-algorithms-to-enhance-neural-networks-cde902384b6e