

Zone-Based Path Planning of a Mobile Robot Using Genetic Algorithm



**B. G. Sumanth Bhaskar, Amit Rauniyar, Rahul Nath,
and Pranab K. Muhuri**

Abstract Path planning is one of the widely studied problems in mobile robotics, which deals in finding an optimal path for a robot. To generate a collision-free path for a robot in an environment by satisfying certain constraints is a complex task. So, path planning is an NP-hard problem. In this paper, we present a new formulation of the path planning problem for a mobile robot by introducing zones which are neighbors to the static obstacles, through which a robot can pass avoiding the collisions. Our proposed model has an advantage of shrinking the search space which in turns reduces the computational complexity. We consider the minimization of the travel distance of a robot as the main objective to find a feasible path. We implement a genetic algorithm (GA) as a solution technique and compare it with two other well-studied meta-heuristic algorithms, viz. Tabu Search and Simulated Annealing. Further, we incorporate a modified mutation operation in all three algorithms to replace a zone from the reduced search space to generate a new potential solution. The simulations for different environments and comparative analysis using obtained results show that GA performs better than the other two approaches.

Keywords Mobile robots · Path planning · Genetic algorithm · Collision avoidance

B. G. Sumanth Bhaskar (✉) · A. Rauniyar · R. Nath · P. K. Muhuri
Department of Computer Science, South Asian University, New Delhi 110021, India
e-mail: gopisumanth@outlook.com

A. Rauniyar
e-mail: amitrauniyar90@gmail.com

R. Nath
e-mail: rahul.nath@outlook.com

P. K. Muhuri
e-mail: pranabmuhuri@cs.sau.ac.in

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2021

A. Chakrabarti and M. Arora (eds.), *Industry 4.0 and Advanced Manufacturing*,
Lecture Notes in Mechanical Engineering,
https://doi.org/10.1007/978-981-15-5689-0_23

1 Introduction

With the advancement in robotics, mobile robots are employed in several domains like manufacturing, assembling, space exploration, nuclear power plants, and mobile systems. Path planning is one of the critical issues in the field of mobile robots, which has attracted many researchers to design efficient solution techniques [1]. Here, the primary goal is to find the shortest feasible path for a robot, avoiding collisions with the obstacles in an environment. Therefore, it can be modeled as an optimization problem subject to certain constraints. The problem becomes very complex with an increase in the size of the environment which may lead to the formation of several possible paths. Thus, a collision-free path planning problem belongs to the category of NP-hard problems [2]. There are many studies on the path planning problem using various techniques such as random trees, roadmaps, cell decomposition, potential fields, and grid-based A* algorithm [3]. Each of these techniques has pros and cons depending upon the type of the domain where it is employed. Most of such methods bear a limitation of finding only one solution, which may not be optimal or shortest even for a static environment [4]. Due to the complexity of the problem, traditional optimization techniques become incompetent and intractable to solve it [2].

Heuristic approaches like evolutionary algorithms (EAs) are proved to be a powerful tool as they are simple to design robust solution methods for path planning problems [5]. As a meta-heuristic technique, genetic algorithms (GAs) have been widely studied for several optimization problems including mobile robot path planning problem [6]. Manikas et al. [1] proposed a GA-based navigation technique for an autonomous robot to traverse the path. An environment with static and dynamic obstacles was designed by Mahjoubi et al. [7] and formulated a solution technique using GA to avoid collisions. Similarly, to design a path for a robot, Song et al. [8] used a grid representation of an environment to simulate their GA-based solution technique along with a Bezier curve. A new binary encoded matrix was developed by Patle et al. [9] for GA to generate a path together with an optimum controller method. A new selection approach including the robot's speed into the encoding of the solution was designed by Wang et al. [10]. They developed a path planning approach using GA to find a near optimal solution avoiding the obstacles. In [4], Tuncer et al. integrated a new mutation operator into a GA-based solution to find an optimal path in a grid-based environment. Despite several studies, the path planning problem still needs to be formulated efficiently to decrease the complexity of search space. Further, it also requires a tool to handle the infeasibility of generated solutions to find a collision-free optimal path for a robot.

In this paper, we formulate the problem of path planning for a robot, creating zones which are neighbors to the corners of the static obstacles in an environment. This helps to reduce the search space to design a path. Our formulation is based on the technique to derive an optimal path, ensuring that a robot travels from its initial point to a destination point through these zones avoiding the collisions. Further, we modify the mutation operation to produce a solution by strictly substituting a different zone into the existing solution from the reduced search space. Thus, we proposed a

GA-based planning method and incorporated the modified mutation operation into it to tackle the premature convergence. For a fair comparison, we also implement two other widely studied nature-inspired algorithms for path planning problem, viz. Simulated Annealing (SA) [11] and Tabu Search (TS) [12] and integrate the modified mutation operation into their basic framework. Through extensive simulations, we show that all the three approaches are able to generate feasible solutions, but the proposed GA-based solution method produces better results with good convergence compared to SA and TS.

Further, the paper is organized into several sections: Sect. 2 describes the problem formulation along with the environment representation. Section 3 explains the design and implementation of evolutionary algorithms for the addressed problem. Section 4 provides the experimental results and comparative analysis. Finally, Sect. 5 concludes the study done in this paper and provides future directions.

2 Problem Formulation

In this section, we describe the real-space representation as an environment which comprises the free space and the obstacles. A suitable environment representation is required to find a feasible path for a robot to reach its destination. Several path planning methods use the grid-based model [4] where the real-space is divided into grids. Such a layout can be characterized as a two-dimensional (2D) coordinate plane which allows straightforward calculation of the distance between points connecting two grids. Therefore, we have adapted this model as it can simulate real-world scenarios and are easy to understand. Here, we formally define the mathematical framework where the robot has the knowledge about the environment. In the grid-based model, the whole space is divided into t number of grids which can be expressed as

$$G = \{g_i | \forall 0 \leq i \leq t - 1\} \tag{1}$$

where g_i is the grid identifier. The grids occupied by the obstacle can be denoted as O , where O is the subset of G . Therefore, the obstacle-free grids $F = G - O$ from which Z zones are identified to find the feasible path without any obstacles. This finally reduces the search space.

An example is shown in Fig. 1 which illustrates the grid model used for our study. Here, the environment is divided into 10×10 grids. Therefore, the environment consists of 100 grids in total. Each grid is having a length and breadth of 1 unit. The grids in black color are the static obstacles, and the rest of the grids are free grids where a robot is free to move. From the set of free grids, we create different subsets of colored zones which are neighbor grids of the obstacles' corner as shown in Fig. 2. Altogether we identified 12 zones which consist of 36 colored grids, i.e., 36% of the G and 56% of F . This results into reduced search space and less computation time to generate a feasible and optimized path. For a robot where there are obstacles between

Fig. 1 Example 10×10 grid-based environment

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

Fig. 2 Example 10×10 grid-based environment with zones

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

the start position and the goal position, the path is the one closer to the corners of the obstacle.

Further, we have adapted the coordinate system model for the environment which enables to calculate the Euclidian distance and check the feasibility of the path easily. For the grid environment in Fig. 2, the top left corner of the grid g_0 is the origin. We assume the robot travels through the midpoint of each grid. The coordinates of g_0 are $(0.5, 0.5)$, g_1 are $(1.5, 0.5)$, g_2 are $(2.5, 0.5)$, and so on. Assuming $g_0 (0.5, 0.5)$ is the starting point for the robot and the destination is $g_{99} (9.5, 9.5)$. The X and Y coordinates of a grid can be calculated as follows:

$$X_i = \text{modulus}(g_i, 10) + 0.5 \tag{2}$$

$$\begin{aligned}
 Y_i &= \text{floor}(g_i/10) + 0.5 \\
 0 \leq i &\leq t - 1
 \end{aligned}
 \tag{3}$$

For a mobile robot, we need to find the shortest path between its starting and destination point. Therefore, the optimization problem becomes the minimization of distance to be travelled by the robot avoiding the obstacles. There is a possibility that paths generated may have obstacles. In order to compute the distance of these infeasible paths, we impose a penalty based on the number of obstacles on those paths. As a result, these paths will have a larger distance. The distance function [4] can be formulated as

$$D = \begin{cases} \sum_{b=1}^{n-1} \text{distance}(g_b, g_{b+1}) & \text{for feasible paths} \\ \sum_{b=1}^{n-1} \text{distance}(g_b, g_{b+1}) + \text{penalty} & \text{for infeasible paths} \end{cases}
 \tag{4}$$

where a and b are the two consecutive grids in the path and

$$\text{distance}(g_b, g_{b+1}) = \sqrt{(x_b - x_{b+1})^2 + (y_b - y_{b+1})^2}
 \tag{5}$$

Hence, the objective function can be characterized as:

$$\text{Min}\{D\}
 \tag{6}$$

where D is the total sum of the distance between all connecting grids of the path. Coordinates of the obstacles are used to generate equations of their edges which help in checking the feasibility of the path. The determinant is calculated between the equation of the line joining the grids and the equation of the edges of obstacles. The obtained determinant value concludes if there is any intersection between these lines. This helps to determine the feasibility of the path [13, 14].

3 Solution Techniques

We have implemented our solution techniques using three widely studied nature-inspired approaches, namely GA [4, 15, 16, 17], SA [11, 18] and TS [12, 19] for a fair comparison. The standard framework of each algorithm is used for the experiments. Further, we have integrated a modified mutation operation into each of these algorithms to generate a new potential solution. The pseudocodes for working of GA, SA, and TS developed for the proposed problem model are mentioned in Algorithm 1, Algorithm 2, and Algorithm 3, respectively. Now, we discuss the basic components of the algorithms.

Algorithm 1: The framework of GA

Input: G, O, Z , population size, maximum iterations

Step 1: **Population Initialization:** initialize a set of individuals of fixed length

Step 2: **while** termination criterion is not met **do**

i. **Fitness evaluation** of the individuals using the objective function.

ii. **Selection:** the binary tournament selection is used

iii. **Crossover:** PMX technique is used for crossover

iv. **Modified mutation:** see Algorithm 4

v. Fitness is calculated for the newly generated population

end while

Output: the feasible path

Algorithm 2: The framework for SA

Input: G, O, Z , initial temperature, population size, maximum iterations

Step 1: **Population Initialization:** Randomly initialize a set of individuals of fixed length

Step 2: **Fitness evaluation** of the individuals

Step 3: **Best solution:** Select the individual with the best fitness

Step 4: Set Temperature to the initial temperature

Step 5: **while** termination criterion is not met **do**

i. **Create neighbour:** Modified mutation is used to create the neighbour

ii. Evaluate the fitness of generated individuals

iii. Sort population

iv. **Best solution:** Select the individual with the best fitness

v. update the temperature

end while

output: feasible path

Algorithm 3: The framework for TS

Input: G, O, Z , initial temperature, population size, maximum iterations

Step 1: **Seed generation:** randomly generate an individual of a fixed length and store it as the best solution

Step 2: **while** termination criterion is not met **do**

i. Generate neighbours to the existing individual

ii. Evaluate the fitness of all individuals

iii. Update the tabu list

iv. Select the best individual among all

end while

output: feasible path

0	31	62	66	99
---	----	----	----	----

Fig. 3 Encoding of an individual

3.1 Encoding of an Individual

All of the above three algorithms considered for our experiment begin with the encoding of an individual. An individual is a candidate solution for the path planning problem. Non-negative integers are used for the encoding of an individual as it takes less memory and less space in optimization than the other encoding techniques [10]. A fixed length of the individual is generated which comprises the grid labels to represent the path to be travelled by the mobile robot. The elements of an individual consist of a grid from where the path begins, the grids from the zones and the destination grid where the path ends. For example, Fig. 3 demonstrates an individual (for the 10×10 grid of Fig. 1) where the path begins at g_0 and g_{31} , g_{62} , g_{63} are zones through which the path is traced, and g_{99} is the end point of path. Here, the position of g_0 and g_{99} is fixed at the beginning and end of individual, respectively.

As GA and SA are population-based technique, we generate a fixed size of the initial population of individuals for the experiments. The pool of individual represents a variety of solutions. The generated population may contain both feasible and infeasible paths.

3.2 Fitness Evaluation

We evaluate the fitness of each individual using the objective function defined in Eq. 6 for all three algorithms. We calculate the fitness value as the total distance for feasible paths, and for infeasible paths, we take summation of total distance with penalty, respectively. The penalty must be greater than the maximum possible distance. Therefore, for our experiments, we took a penalty of 100 units. If there is an infeasible path between any two elements of an individual, we add 100 every time.

3.3 Selection for GA

We have used the binary tournament selection method which selects the best candidate solutions and passes them to the next phase of GA. The idea is to select a better individual and transfer them to the next generations [20].

3.4 Crossover for GA

The crossover operator is only employed for GA to recombine two selected individuals to create new solutions. For permutation encoding, partially mapped crossover (PMX) is widely used as it generates an individual without any repetition. This technique helps the robot not to travel through an already travelled grid again. In PMX, two crossover points are selected randomly, and respective elements are exchanged in such a way that there are no repeated elements in the individual [20].

3.5 Modified Mutation for GA and SA

To generate a new solution, we have developed a modified variant of single-point mutation [15] to ensure that a new grid is chosen specifically from the list of available zones Z . Algorithm 4 provides the steps followed for modified mutation for our experiments. An individual is selected randomly from the population. Also, a mutation point is chosen randomly. Then, a grid of the individual is replaced with a new grid from the zones without any repetition. This is employed to GA and SA only during our experiments. This operation helps to avoid premature convergence and increase the diversity of the population.

Algorithm 4: *modified_mutation (individual, Z)*

begin

Select an individual I randomly

A $mutation_{point}$ is chosen randomly from I

Select a new grid g_k from $Z - I$ randomly

Replace the value at $mutation_{point}$ with g_k

end

4 Experiments and Results

This section describes the experimental setup and analyzes the results obtained. We have used two different sized datasets for our experiments, one with 10×10 grids (see Fig. 2) and other with 20×20 grids (see Fig. 4). We have implemented GA, SA, and TS as a solution technique to solve the proposed model. We take a population size of 10 for both GA and SA to experiment with the datasets. For GA, we use tournament selection with tournament size 2 and set the probability of crossover to 0.5. Mutation plays a vital role in generating better solutions and helps to escape out of local minima. Therefore, we set the probability of mutation in GA and mutation rate in SA to 0.8. Further, for SA, we set the values of a number of sub-iterations,



Fig. 4 20 × 20 grid-based environment

number of neighbors, initial temperature, and temperature reduction rate to 5, 3, 0.1, and 0.99. We keep the standard parameter settings for TS [12, 19]. The termination criterion for all three algorithms is kept as 100 maximum iterations.

We have conducted 50 independent runs of each algorithm using both datasets. We recorded the objective values of the best individual found at each generation. We also note the running time taken by each algorithm to its complete execution. In order to evaluate the efficiency of algorithms in terms of solutions generated, we have calculated the variance over the obtained results. For comparative study, the average values of 50 runs of each algorithm are summarized in Table 1 for 10 × 10 and 20 × 20 datasets. We can witness that all the three algorithms have been able to generate feasible and varied solutions. It can be observed that the GA-based solution approach has produced much better results compared to SA and TS. Figures 5 and 6 present the graph plot of average fitness versus iteration for 10 × 10 and 20 × 20

Table 1 Comparative results for 10 × 10 and 20 × 20 grids

10 × 10				20 × 20		
Algorithms	Best fitness	Variance	Running time	Best fitness	Variance	Running time
GA	14.204	0.0249	2.2555	29.451	0.0183	5.1117
SA	14.724	0.1703	6.2706	29.883	0.1141	15.8222
TS	15.68	3.5309	0.3612	30.682	0.5838	1.2295

Fig. 5 Average best fitness over varying iterations for 10×10

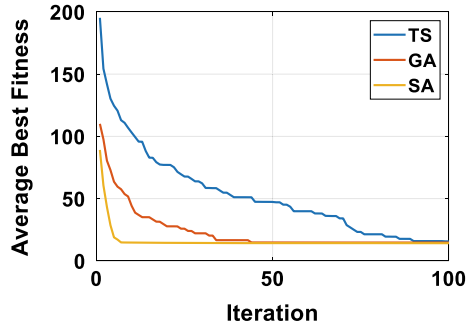
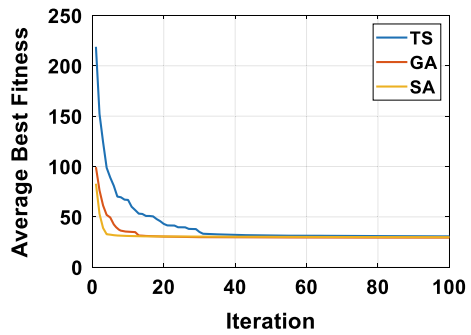


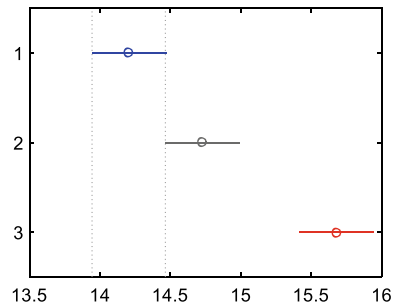
Fig. 6 Average best fitness over varying iterations for 20×20



datasets, respectively. The convergence of GA is much better and faster compared to SA and TS. Thus, we can conclude that GA has outperformed SA and TS in terms of solution generations.

To evaluate the significant difference, we have also conducted an ANOVA test with a confidence level of 95% using the best fitness values obtained at the last generation of each 50 runs of the algorithms. The graphs for comparison of significance difference of means for 10×10 and 20×20 are represented in Figs. 7 and 8, respectively. Here, GA, SA, and TS are represented by 1, 2, and 3, respectively, on Y-axis of the graphs. From Fig. 7, we can see that there is no significance difference in the means

Fig. 7 Comparative results of ANOVA test for 10×10 using GA, SA, and TS



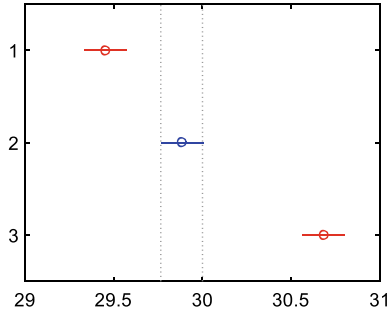


Fig. 8 Comparative results of ANOVA test for 20 × 20 using GA, SA, and TS

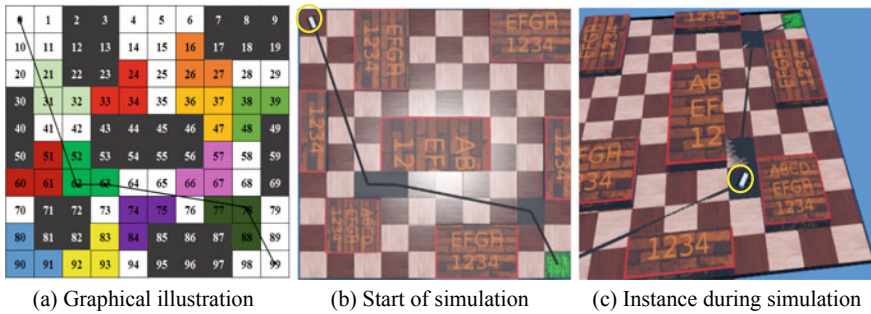


Fig. 9 Generated solution for 10 × 10 grid environment simulated on Webots for GA

of GA and SA. On the other hand, mean of TS is significantly different and larger than the means of GA and SA. From the analytical study of Fig. 8, we can conclude that the means of all three algorithms are significantly different. Here, again TS has produced a solution with larger variation and poor fitness values compared to GA and SA. For both of the datasets, GA has been able to generate a better solution with good convergence. We further simulated both the environments in Figs. 2 and 4 on Webots R2019a version 2.0 simulator for a six-wheeled robot. In Fig. 9, the leftmost image illustrates a feasible path generated by GA, and the two screenshots on the right demonstrate the simulation of the obtained path for 10 × 10 grid environment. The video of this simulation can be seen at the link provided in [21]. Here, the brown boxes with alphanumeric characters are obstacles, black grids are the zones given by GA, and green grid is the target position.

5 Conclusion

In this study, we have proposed a technique to design the path between grids for a robot using the zones. This helps to reduce the search space and find an optimal

path for a robot, avoiding the collisions with the static obstacles. In order to increase diversity, we have incorporated a modified mutation technique to ensure a selection of grids from the available zones without any repetition. For a fair comparison, we have implemented three different solution techniques using GA, SA, and TS and check the efficiency considering two differently sized datasets. The comparative analysis and significance test of the algorithms are done using the results obtained. All three algorithms have been able to generate feasible solutions, but the GA-based technique has shown better performance in terms of solution generations. We have also performed the simulation of solutions obtained in Webots simulator using a mobile robot. Further, we will design an environment for a manufacturing process using multiple mobile robots and identify other critical objectives of the problem. We will also formulate the multi-objective path planning problem and solve it with other multi-objective EAs.

References

1. Manikas, T. W., Ashenayi, K., & Wainwright, R. L. (2007). GAs for autonomous robot navigation. *IEEE Instrumentation and Measurement Magazine*, 10(6), 26–31.
2. Patle, B. K., Pandey, A., Jagadeesh, A., & Parhi, D. R. (2018). Path planning in uncertain environment by using firefly algorithm. *Defence Technology*, 14(6), 691–701.
3. Barraquand, J., Langlois, B., & Latombe, J. C. (1992). Numerical potential field techniques for robot path planning. *IEEE Transaction on Systems, Man, and Cybernetics*, 22(2), 224–241.
4. Tuncer, A., & Yildirim, M. (2012). Dynamic path planning of mobile robots with improved genetic algorithm. *Computers & Electrical Engineering*, 38(6), 1564–1572.
5. Mac, T. T., Copot, C., Tran, D. T., & De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, 86, 13–28.
6. Kotthoff, L. (2016). Algorithm selection for combinatorial search problems: A survey. In *Data Mining and Constraint Programming* (pp. 149–190). Cham: Springer.
7. Mahjoubi, H., Bahrami, F., & Lucas, C. (2006, July). Path planning in an environment with static and dynamic obstacles using genetic algorithm: A simplified search space approach. In *2006 IEEE International Conference on Evolutionary Computation* (pp. 2483–2489). IEEE.
8. Song, B., Wang, Z., & Sheng, L. (2016). A new genetic algorithm approach to smooth path planning for mobile robots. *Assembly Automation*, 36(2), 138–145.
9. Patle, B. K., Parhi, D. R. K., Jagadeesh, A., & Kashyap, S. K. (2018). Matrix-Binary Codes based Genetic Algorithm for path planning of mobile robot. *Computers & Electrical Engineering*, 67, 708–728.
10. Wang, Y., Sillitoe, I. P., & Mulvaney, D. J. (2007). Mobile robot path planning in dynamic environments. In *Proceedings 2007 IEEE International Conference on Robotics and Automation* (pp. 71–76). IEEE.
11. Miao, H., & Tian, Y. C. (2013). Dynamic robot path planning using an enhanced simulated annealing approach. *Applied Mathematics and Computation*, 222, 420–437.
12. Yoshikawa, M., & Otani, K. (2010). Ant colony optimization routing algorithm with tabu search. In *Proceedings of the International Multiconference of Engineers and Computer Scientists* (Vol. 3, pp. 17–19).
13. Antonio, F. (1992). Faster line segment intersection. In *Graphics Gems III (IBM Version)* (pp. 199–202). Morgan Kaufmann.
14. Phelps, A. M., & Cloutier, A. S. (2003). *Methodologies for quick approximation of 2D collision detection using polygon armatures* (Vol. 14623). Rochester, NY: Rochester Institute of Technology.

15. Muhuri, P. K., Rauniyar, A., & Nath, R. (2019). On arrival scheduling of real-time precedence constrained tasks on multi-processor systems using genetic algorithm. *Future Generation Computer Systems*, 93, 702–726.
16. Goldberg, D. E. (2006). *Genetic algorithms*. Pearson Education India.
17. Nolfi, S., Floreano, D., & Floreano, D. D. (2000). *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. Cambridge: MIT Press.
18. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by Simulated Annealing Science*, 220(4598), 671–680.
19. Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093–2229). Boston, MA: Springer.
20. Yuan, S., Skinner, B., Huang, S., & Liu, D. (2013). A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *European Journal of Operational Research*, 228(1), 72–82.
21. <https://www.youtube.com/playlist?list=PLDcamukFWEQQ6W8UmzjgnO4OjUoJcV7i0>.