An Improved Genetic Algorithm Based Robot Path Planning

A thesis

submitted in partial fulfillment of the requirement for the Degree of

Master of Computer Science and Engineering

of

Jadavpur University

By

Ritam Sarkar

Registration No.: 128988 of 2014-15

Examination Roll No.: M4CSE1601

Under the esteemed Guidance of

Prof. Nirmalya Chowdhury

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

India

2016

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

<u>Certificate of Recommendation</u>

This is to certify that the dissertation entitled "An Improved Genetic Algorithm Based Robot Path Planning" has been carried out by Ritam Sarkar (University Registration No.: 128988of 2014-15, Examination Roll No.: M4CSE1601) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Computer Science and Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other University or Institute.

…………………………………………………

Prof. Nirmalya Chowdhury(Thesis Supervisor)
Department of Computer Science and Engineering
Jadavpur University, Kolkata-32

Countersigned

…………………………………………………….

Prof. Debesh Kumar Das
Head, Department of Computer Science and Engineering,
Jadavpur University, Kolkata-32.

…………………………………………………….

Prof. Sivaji Bandyopadhyay
Dean, Faculty of Engineering and Technology,
Jadavpur University, Kolkata-32.

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

<u>Certificate of Approval*</u>

This is to certify that the thesis entitled “An Improved Genetic Algorithm Based Robot Path Planning” is a bona-fide record of work carried out by Ritam Sarkar in partial fulfillment of the requirements for the award of the degree of Master of Computer Science and Engineering in the Department of Computer Science and Engineering, Jadavpur University during the period of June 2015 to May 2016. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

……………………………………………………………………………..

Signature of Examiner 1

Date:

……………………………………………………………………………..

Signature of Examiner 2

Date:

*Only in case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

<u>Declaration of Originality and Compliance of Academic Ethics</u>

I hereby declare that this thesis entitled "An Improved Genetic Algorithm Based Robot Path Planning" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Computer Science & Engineering.

All information have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Ritam Sarkar
Registration No: 128988 of 2014-15
Exam Roll No.: M4CSE1601
Thesis Title: An Improved Genetic Algorithm Based Robot Path Planning

…..……………………………………….

Signature with Date

Acknowledgement

I express my sincere gratitude to **Prof. Nirmalya Chowdhury**, my guide for his affectionate and valuable guidance without whose help the present work could not have been a successful one. I am also indebted to him as a Professor who introduced me to the world of Data Mining and Pattern Recognition.

Also I thank **Prof. Debesh Kumar Das**, Head of the Department of Computer Science and Engineering, for his assistance in allowing me to work in the departmental laboratory without which my work would have been incomplete.

I would also like to convey my sincere gratitude to all my respected teachers and faculty members in this department for their invaluable suggestions and kind cooperation.

I express my thanks to all friends of my class.

Last but not the least, the encouragement given by my mother **Mrs. Kalyani Sarkar**, my father **Mr. Bhola Nath Sarkar** who have always been a constant source of inspiration and whose encouragement is beyond linguistic expression for me.

…………………………………………………..

Ritam Sarkar

Registration No: 128988 of 2014-15

Exam Roll No.: M4CSE1601

Department of Computer Science & Engineering

Jadavpur University

Contents

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction to robotics

1.1 Introduction

Russian-born American science-fiction writer Isaac Asimov first used the word “robotics” in 1942 in his short story "Runabout." Asimov had a much brighter and more optimistic opinion of the robot's role in human society than did Capek. Asimov also proposed three "Laws of Robotics" that his robots, as well as sci-fi robotic characters of many other stories, These are: 1) A robot may not injure a human being or, through inaction, allow a human being to come to harm. 2) A robot must obey the orders given it by human beings except where such orders would conflict with the First Law. 3) A robot must protect its own existence as long as such protection does not conflict with the First or Second Law. The word robotics was derived from the word robot, which was introduced in a play about mechanical men that are built to work on factory assembly lines and that rebel against their human masters. These machines in *R.U.R.* (*Rossum's Universal Robots*), written by Czech playwright Karl Capek in 1921, got their name from the Czech word for slave [1]. The word robot comes from the Slavic word robota, which means labour. A robot is a mechanical or virtual artificial agent, usually an electro-mechanical machine that is guided by a computer program or electronic circuitry [2].

Robotics is the branch of mechanical engineering, electrical engineering and computer science that deals with the design, construction, operation, and application of robots as well as computer systems for their control, sensory feedback, and information processing [2]. Mechanical engineering responsible for machinery and structure of the robot whereas electrical engineering includes controlling and intelligence of the robot and finally the computer science deals with the movement and the observation of robots.

1.2 Different aspects of robotics

Though the origin of robotics lie in the far history, but research into the functionality and potential uses of robots did not grow substantially until the 20th century. Throughout history, it has been frequently assumed that robots will one day be able to mimic human behavior and manage tasks in a human-like fashion. Today, robotics is a rapidly growing field, as technological advances continue; researching, designing, and building new robots serve various practical purposes. As a result of that there exist numerous aspects of robotics [3]. Some of them are described below.

i. Adaptive control

Adaptive control is the process of controlling a system adaptively as the parameter changes. Such as, while an aero plane flies its mass reduces gradually due to fuel consumption, that's why a control law is needed to control the system adaptively according to the changes in the parameters.

ii. Unmanned aerial vehicle (UAV)

Unmanned aerial vehicle (UAV), also known as drone is a kind of aircraft which are not conducted by any human pilot riding on it. These are controlled either by remote from the ground or in another vehicle or may be totally automatic. UAVs are generally used in military for special operations where any manned aerial vehicle could be dangerous.

iii. Bio-inspired robotics

Bio-inspired is a field of robotics that is inspired from the biological system where biological knowledge are applied to the engineering problem. Bio-inspired robotics is different from bio-mimicry. Bio-mimicry is just the way of copying the nature, is a different branch of robotics, called soft robotics whereas Bio-inspired robotics is the way of learning and designing mechanisms that are easier to implement as well as more effective than the nature.

iv. Autonomous car

Autonomous car is a kind of vehicle which can drive itself without the help of any human being. For that purpose, it has to detect as well as discriminate all the surroundings with the help of various technologies like radar, GPS, Computer vision etc.

v. Cloud Robotics

Cloud robotics is the field of robotics where a robot attempts to invoke cloud technologies like cloud computing, cloud storage etc. As a result, when the robots are able to establish connection with the cloud, they get a lot resources from the cloud like storage, powerful computation etc. which can process and share information from various robots. Some examples of cloud robotics are Google's self-driving car, cloud medical robot etc.

vi. BEAM robotics

BEAM robotics is a combination of biology, electronics, aesthetics and mechanics. It is a kind of robotics that is based on simple analogue circuits instead of microprocessor. As a consequence, it is less flexible but more robust and efficient than microprocessor based robotics.

vii. Cognitive robotics

Cognitive robotics are different from regular Industrial robotics where robots are programmed to do just a particular type of work, on the contrary in Cognitive robotics, the robots are allowed to learn and reason about how to behave in response to complex situations. Cognitive robotics is basically part of robotics which deals with cognitive phenomena like anticipation, perception, learning, reasoning etc.

viii. Swarm robotics

Swarm robotics is based on the co-ordination of multi-robot which concentrates on the collective behavior which is emerged from the interaction between the robot and environment.

ix. Motion planning

Motion planning is a section of robotics which is responsible for robot navigation system where the objective is to build an optimal or near optimal path for the robot by avoiding the obstacle in very complex environment.

x. Human Computer Interaction (HCI)

The term HCI first used in [4], it includes the design and use of computer technology by focusing on the interfaces between users and computers, where users interact with computers and design technologies that let users interact with computers in novel ways.

1.3 An overview of robot path planning problem

Robot’s path planning has been one of the important research topic from past few years. In a Robot’s path planning problem, each possible path is associated with a start node, target node and an environment through which the Robot traverses. The aim is to get an optimal or near optimal path considering distance traverse by the robot being the optimization criteria. Robot’s environments consist of a number of obstacles. Depending upon the nature of obstacles, Robot’s path planning is of two types: 1. Robot’s path planning in an environment where the obstacles are fixed throughout the whole process which is known as a static environment. 2. Robot’s path planning in an environment where the obstacles are not fixed. For instance the obstacles may change their positions and / or some new obstacles may appear / disappear as time goes on. This is called dynamic environment. Robot path planning can also be broadly divided into two sections based on the availability of the prior knowledge about the environment. Those are off-line path planning and on-line path planning. Off-line is a kind of path planning where the prior knowledge about the environment is known in advance where as in case of on-line path planning the prior information regarding the environment is unknown, the robot gets the information through sensors whenever there is any changes in the environment.

Path planning in real world scenario is a kind of cycle which constitutes of a few important steps. Those are perception, map building, cognition map control and motion control as shown in the figure 1 [6]. Perception is responsible for extracting and interpreting essential data for the purpose of building map through some sensors from the real world environment. Then cognition map control, for the motion control of the robot. Then again sensing the environment in order to gather information regarding any changes occur in the environment and modify the map accordingly and so on [5, 6].

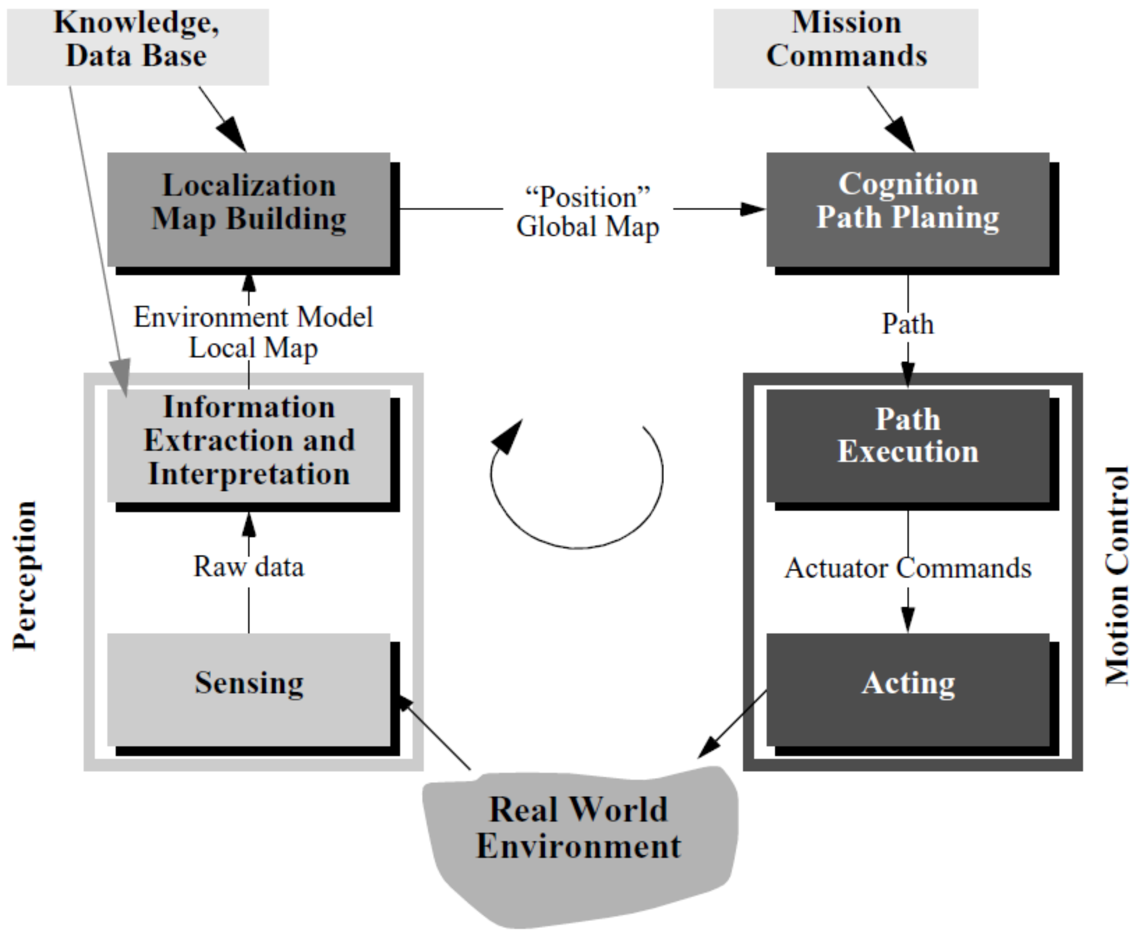


Figure 1:- An graphical overview of path planning problem
Image courtesy: Reference [6]

1.3.1 Representation of environment

Representation of environment has a great role in robot path planning problem. Environment representations can be broadly divided into two approaches [7]. These are discrete approximation approach and continuous approximation approach. According to discrete approximation, the environment is discretized into a number of equal shaped and sized grids. In case of continuous approximation, every chunk of the environment is considered as to a vertex, which are connected by edges, if a robot can navigate from one vertex to the other. The environment can also be further represented by a variety of ways [8]. Some of them are described below.

i. Classical exact cellular decomposition

According to this, the free regions of the environment are divided into non-overlapping regions, called cells which are zigzag shaped. In this type of environment representation, path planning is consist of two steps. In the first step, the division of free spaces into cells and store them as adjacency graph where each node represents a cell and the edge represents the relationship between the cells. Then in second step, the planner evaluates the path as a walk through the adjacency graph [8 – 11]. Some examples of classical exact cellular decomposition are trapezoidal decomposition, boustrophedon decomposition. In trapezoidal decomposition [12, 13] the cells are trapezoids whereas in boustrophedon decomposition [14], the number of cells are less thereby reducing the coverage path than the trapezoidal decomposition.

ii. Morse based cellular decomposition

Unlike exact cellular decomposition, morse based cellular decomposition can deals with non-polygonal obstacles. By using different morse function, different shaped grids can be obtained, like circular, spiked cells etc. After completion of cell decomposition, a path associated with adjacency graph is obtained through the planner [8]. Boustrophedon is also a type of morse based cellular decomposition [11]. Sensor based path coverage can be done by using on-line morse based boustrophedon decomposition through the sensor data [15, 16].

iii. Grid based decomposition

In grid based decomposition the environment is decomposed into a number of grids which are equal in size and shape. Each grid represents either some free space or obstacle. Grid based decomposition has a lot of benefits. Such as, it is very convenient to map the environment and the information of each grids can be stored easily using an array. But grid based decomposition introduces some extra usages of memory as the size of the environment increases and also fails to recognize the narrow spaces between the obstacles. There exist a lot of algorithms for robot path planning problem which are mostly based on grid based decomposition. These are wave front algorithm, neural networks, GAs etc.

iv. Graph based representation

In graph based techniques, the environment is mapped into a graph consist of a number of nodes and the edges connecting the nodes. This approach may have some issues [8, 17]. 1) The prior knowledge regarding the environment might be insufficient. 2) It may contain some constraint on the robot motion, like the robot may traverses only on one way for some part of the environment. 3) Getting real-time knowledge through the sensor and modify the information accordingly.

Chapter 2: Introduction to soft computing

2.1 Introduction

As the name suggests the term “soft computing” is just the opposite concept of “hard computing”. All the classical reasoning and modeling approaches that are based on Boolean logic, analytical models and crisp classifications fall in hard computing category. Hard computing comes down hard on precision leaving no room for approximations. This can be computationally expensive, time consuming and sometimes even impossible for application to complex real-life problems because many such problems are typically ill-defined systems, difficult to model with large solution spaces. On the contrary, soft computing deals with these type of problems with the same way as human deals with them, i.e. on the basis of intelligence, common sense, consideration of analogies, approaches etc. Unlike hard computing, soft computing methods are the based on approximation. Another important difference between hard computing and soft computing is that soft computing is based on intelligence whereas hard computing is based on intelligent system [73].

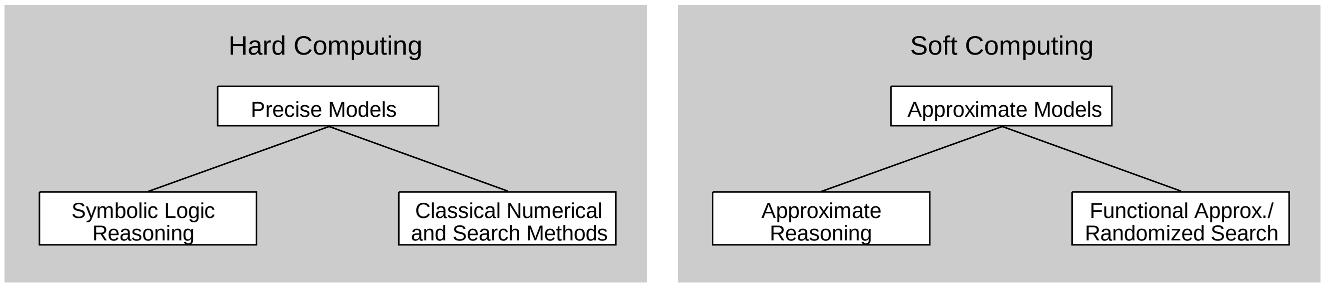


Figure 2: Graphical summary of the domains covered by hard computing and soft computing
Image courtesy: reference [18]

The term Soft computing was proposed by the inventor of fuzzy logic [19]. He describes it as follows:

“Soft computing is a collection of methodologies that aim to exploit the tolerance for imprecision and uncertainty to achieve tractability, robustness, and low solution cost. Its principal constituents are fuzzy logic, neurocomputing, and probabilistic reasoning. Soft computing is likely to play an increasingly important role in many application areas, including software engineering. The role model for soft computing is the human mind.”

2.1 Three principle components of Soft Computing

Three principle components of soft computing are Fuzzy logic, Artificial Neural Network and Evolutionary Computing. These are described below.

2.2.1 Fuzzy Logic

The term fuzzy logic was introduced with the proposal of fuzzy set theory by Lotfi Zadeh [20], a professor at the University of California at Berkley. Fuzzy Logic is a method of reasoning that resembles human reasoning. Fuzzy Logic imitates the way of decision making in humans that involves all intermediate possibilities between digital values TRUE and FALSE. The conventional logic block i.e. a computer takes precise input and produces a definite output as TRUE or FALSE. On the contrary, Fuzzy logic is based on the human decision making techniques that includes a range of possibilities between TRUE and FALSE. Such as TRUE with 100% probability, TRUE with some probability, TRUE and FALSE having 50% probability each, FALSE with some probability and FALSE with 100% probability. Professor Zadeh reasoned that people does not require precise, numerical information input, and yet they are capable of highly adaptive control. If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement. Unfortunately, U.S. manufacturers have not been so quick to embrace this technology while the Europeans and Japanese have been aggressively building real products around it. Some of the usefulness of using Fuzzy Logic includes It may not give accurate reasoning, but acceptable reasoning. Fuzzy logic helps to deal with the uncertainty in engineering. Humans and animals often operate using fuzzy evaluations in many everyday situations. In the case where someone is tossing an object into a container from a distance, the person does not compute exact values for the object weight, density, distance, direction, container height and width, and air resistance to determine the force and angle to toss the object. Instead the person instinctively applies quick “fuzzy” estimates, based upon previous experience, to determine what output values of force, direction and vertical angle to use to make the toss [21].

2.2.2 Artificial Neural Network

Artificial Neural Networks (ANN), first explored by Rosenblatt [22], Widro and Hoff [23] which are basically paralleled distributed information processing paradigm, inspired by biological nervous system. ANN are generally used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected nodes (neurons) which exchange messages between each other with some numeric weights that can be tuned based on experience. The picture of artificial neural network has been shown in the Figure 3 where each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another. As shown in the figure 3, there are three Layers of Neural Networks. They are Input layer, Hidden layer and output layer. The Input layer represents the raw information that is fed into the network. Hidden unit is determined by the activities of the Input layers and the weights on the connections between the input and the hidden layers. The number of Hidden layers may be more than one depending upon the problem scenario on which it is applied and finally the output layer depends on the activity of the hidden layer and the weights between the hidden and output layer.

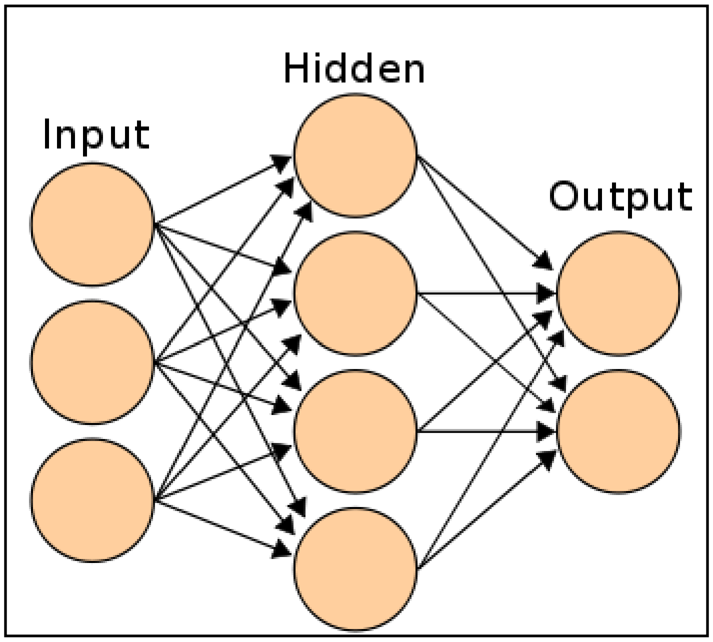


Figure 3: Artificial Neural Network

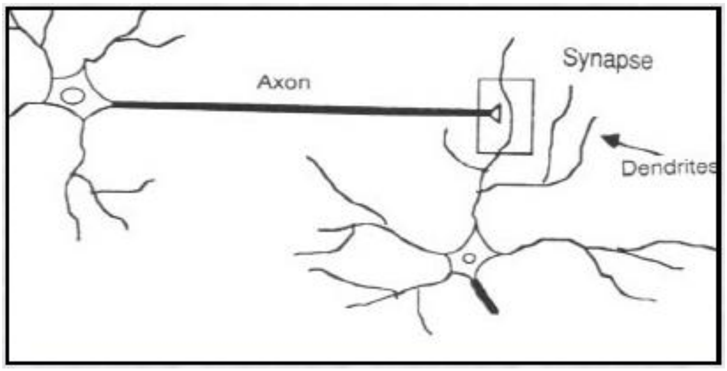


Figure 4: Prototype nerve cell
Image Courtesy - https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

A prototype of nerve cell called neuron as shown in figure 4. The human brain is composed of 100 billion neurons. They are connected to other thousand cells by axons. Electrical impulses propagating along the axon to activate the synaptic junctions. These, in turn, produce further excitations (post synaptic potentials) which travel along the dendrites towards the next neuron. The firing rate of each neuron is controlled by the region where the axon joins the cell body, called the hillock zone. When the membrane potential at the hillock zone rises above a certain threshold value -60mv, it causes a travelling wave of charge to propagate. The neuron must restore itself to its proper resting state of balance before sending out the next packet of charge, called the refractory period. So information is passed via synapses. The synapses are termed excitatory or inhibitory depending on whether the post-synaptic potentials increase or reduce the hillock potential, enhancing or reducing the likelihood of triggering an impulse there respectively. The current level of understanding of the brain function is so primitive that not even one area of brain is yet un-understood. Thus artificial network only tries to mimic the biological neural network in a very crude and primitive manner. The neuron model has been shown in the Figure 5.

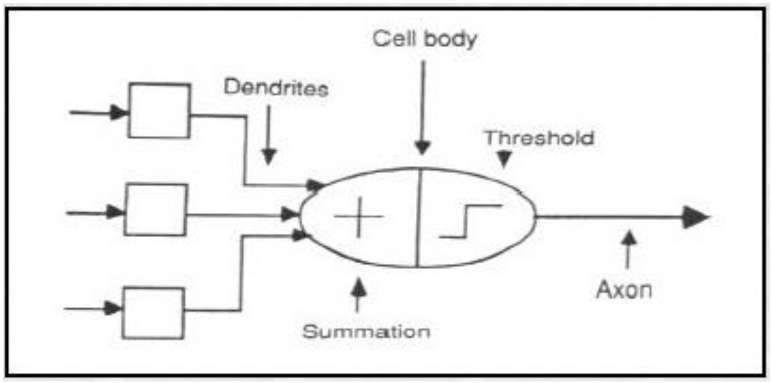


Figure 5: Neuron model
Image Courtesy - https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

There are two types of neural networks in terms of the direction of flow of information. These are feed-forward networks and feedback networks. In feed-forward networks the information flow is unidirectional i.e. from input to output. A unit sends information to other unit from which it does not receive any information and there are no feedback loops whereas the feedback networks contain loop as a result the flow information is bidirectional. Feedback networks are dynamic since their state is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found.

2.2.3 Evolutionary Computing

Evolutionary Computing (EC) is a part of computer science or more specifically is a part of artificial intelligence, inspired by the principle of Darwinian. According to [69], "THE PRINCIPLE OF EVOLUTION is the primary unifying concept of biology, linking every organism together in a historical chain of events. Every creature in the chain is the product of a series of "accidents" that have been sorted out thoroughly under selective pressure from the environment. Over many generations, random variation and natural selection shape the behaviors of individuals and species to fit the demands of their surroundings."

EC is the process of searching in a huge number of possibilities for the solution that allow organisms to survive and reproduce in their environments. In other words, Evolution can also be seen as a method for adapting to changing environments [70]. Now a days EC is being applied to various problems which are even belong to different domains including optimization, automatic programming, signal processing, bioinformatics, social systems, and so on. An important advantage of evolutionary algorithms is that unlike many other optimization techniques, they can cope with multimodal functions [71]. The field of EC techniques can be broadly divided into ant colony optimization, artificial bee colony optimization, artificial immune systems, artificial life, bees algorithm, cultural algorithms, differential evolution, dual-phase evolution, evolutionary algorithms, evolutionary programming, evolution strategy, gene expression programming, genetic algorithms, genetic programming, harmony search, learnable evolution model, learning classifier systems, particle swarm optimization, self-organization such as self-organizing maps [68].

Chapter 3: Literature review

In robot path planning problem there is starting position and a target position and an environment through which the robot traverses in order to find an optimal or near optimal path considering distance traverse by the robot being the optimization criteria. There exist a lot of approaches to this robot path planning problem and also single robot path planning, multi robot path planning, depending upon them it can be divided into a number of sections [5, 8, 24]. Some of them are described below in terms of literature review.

3.1 GAs based approaches

GAs have been used extensively in robot path planning problem. Since it is a multi-dimensional search techniques where it deals with multiple solution simultaneously. The advantages of using GAs are the less computational complexity and the less change to get stuck at the local optimal solution or in other words, avoid premature convergence. Some of GAs based approached are described below.

[1] Yanrong Hu and Simon X. Yang [25] used GAs for their robot path planning problems. They have taken grid based environment where the grids are rectangular in shape. Encoding of chromosome for this type of environment is less time consuming and easy than the other kinds of environments and here chromosomes are considered as orderly numbered grids starting from the start to the target or destination node. Along with the Genetic operators they have also used six knowledge based genetic operators which are only domain knowledge based which incorporate small-scale local search that improves the efficiency of the operators. They have shown experimentally that the genetic operators i.e. the selection, crossover and mutation are not enough for this type of problem, the domain knowledge based operators are also required for getting optimal solution.

[2] Qing Li, Wei Zhang, Yixin Yin, Zhiliang Wang and Guangjun Liu [26] developed an improved genetic algorithm where they have used an obstacle avoidance algorithm for initial population and along with that they have used domain heuristic knowledge based crossover, mutation, refinement and deletion operator for robot path planning. They have also used rectangular grid based environment.

[3] Adem Tuncer and Mehmet Yildirim [27] developed an improved genetic algorithm for dynamic path planning of mobile robot i.e. in an environment where the obstacle are not fixed. They have proposed a new mutation operator which does not produce any infeasible path like other conventional random mutation operator or other modified mutation operators. This mutation operator also avoids premature convergence and very useful in finding the optimal solution.

[4] Amir Hossein Karami and Maryam Hasanzadeh [28] investigated in their adaptive genetic algorithm based robot path planning in 2D complex environments that since robot motion planning problem is generally an NP-hard problem, metaheuristics like GAs are proper way to solve it. They have proposed a novel selection operator which is adaptive in nature. The utility of this selection operator is that the selection probability is updated by using feedback information from the standard deviation of fitness function values in every iteration and it helps to maintain the diversity of the chromosomes in the population in order to overcome the local-trap problem and avoid premature convergence.

[5] Ching-chih Tsai and Hsu-Chih Huang [29] developed a Parallel Elite Genetic Algorithm (PEGA) and its application to global path planning for autonomous robot navigation. The PEGA consists of two Elite Genetic Algorithm (EGA). The searching spaces of both EGAs are not dependent and the populations are also evolved separately for a certain number of generations, called isolation time. After the isolation time a predefined number of subpopulations from the two EGAs, are exchanged through the migration operator. The utility of this migration operator is that it increases the diversity among the individuals of each sub-population and decreases the change to get stuck into a local optima as compared to the conventional GAs. At the end, system-on-a-programmable-chip (SoPcC) based B-spline modeling has been developed in to order to make the path smoother for the robot.

3.2 Swarm Intelligence based approaches

Swarm intelligence also have been used in robot path planning problem. Out of all the existing swarm intelligence approaches, the Particle Swarm Optimization and Ant Colony

Optimization are mainly used to deal with this problem, some of them existing approaches are described below.

[1] Zhang Qiaorong and Gu Guochang [30] developed a path planning based on improved binary particle swarm optimization algorithm. They have used vertex-graph to represent the environment where the shape of the obstacles are taken as polygon. Each path are associated with the start position, vertex of the obstacles and the destination. The length of the particle is equal to the total number of vertices of the obstacles where every bits of the particle are either 0 or 1 in order to represent the vertex is in the path or not. To overcome the limitations of regular binary particle swarm algorithm, they have introduced double-structure particle coding and have also added genetic mutation operator.

[2] Tan Guan-Zheng, He Huan and Solman Aaron [31] presented an Ant Colony System (ACS) algorithm for real-time globally optimal path planning of mobile robots. Their proposed approach has been divided into three steps. In the first step, they have used MAKLINK graph theory to identify the free spaces in the environment where the obstacles are known and polygonal shaped. In the second step, Dijakstra's algorithm has been used to find the sub-optimal path between the starting point and the goal. The path which is obtained by Dijakstra's algorithm are sub-optimal, since they pass through middle point of the free MAKLINK lines. That's why at the last step, ACS has been used to convert the sub-optimal path into the optimal one.

[3] P. Raja and S. Pugazhenthi [32] have developed a particle swarm optimization based techniques for path planning of mobile robot in dynamic environment having the different shaped obstacles, such as convex, concave, curve etc. which are enclosed by rectangular or square boundaries where each path have different length varies from two (i.e. the path having only the start and goal points) to $N + 2$, where N is the total number of vertices of all obstacles in the environment. The effectiveness and efficiency of the proposed algorithm has been demonstrated by the simulation studies.

[4] M. A. Porta Garcia, Oscar Monitel, Oscar Castilo and Roberto Sepulveda [33] developed path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy

cost function evaluation. As the title suggests their approach is based on Simple Ant Colony Optimization Meta-Heuristic (SACO-MH). Their proposed method was named as SACOdm where d is the distance and m represents memory where the decision making is influenced by the existing distance between the source and target node and the ants can remember the visited nodes. Their approach has two operating modes. One is the virtual testing of their proposed path planning algorithm by uploading the map of plain terrain or real terrain. Then the second mode is the planning in the dynamic environment by establishing a connection between mobile robot (MR) and ACO test center (ACOTC). MR will inform the ACOTC about the occurrence of new obstacles through its sensor. In turns, ACOTC will return the coordinate of the optimal path to MR, in order to achieve the coordinated control for tracking the desired path. This approach also use fuzzy cost function to evaluate the best route by fuzzy interface system (FIS). The addition of these features have made the approach ten times faster in finding the optimal collision free path than the regular approaches.

[5] Chengyu Hu, Xiangning Wu, Qingzhong Liang and Yongji Wang [34] introduced a swarm intelligence and stream functions based approach for autonomous robot path planning. The utility of using stream function is that they pushes the robot from the obstacles where all the obstacles are taken as circles. Then PSO is used to generate every optimal forward steps to makeup of the whole path. Though the stream function is easy to implement and simple, but it introduces stagnant point when the stream passing through the obstacles which can be easily eliminated by their proposed PSO.

3.3 Neural Network based approaches

The state space of the Neural Network (NN) is the configuration space of the robot, and the dynamically varying environment is represented by the dynamic activity landscape of the neural network. The target globally attracts the robot in whole state space, while the obstacles locally push the robot away to avoid collisions [24]. Some Neural Network based approaches are described below.

[1] Christopher Kozakiewicz and Masakazu Ejiri [35] developed a neural network based approach to path planning for two dimensional robot motion where the obstacles are polygonal

shaped and can be placed randomly throughout the environment. Their proposed method is based on the camera image feedback loop utilizing neural network (NN) for image processing. The subsequent movement of the robot requires two information. These are the current robot position i.e. the distance and direction from current position to stop position and information about immediate vicinity of the robot. The simulation results suggest that their proposed algorithm is simple and robust as well as computationally efficient.

[2] Fan Jian, Fei NibRui and MA ShiWei [36] introduced a new RL-ART2 neural network based mobile robot path planning. The utility of using ART2 is to store abundant classified pattern size, it is very hard to evaluate and select a large numbers of classified patterns by hand, to eliminate this problem they have introduced reinforcement based learning into ART2. Finally they proposed a collision avoidance system which uses the pattern stored in the ART2 in order to find the optimal or near optimal path.

[3] R. Glasius, A. Komoda and S. Gielen [37] developed a neural network dynamics for path planning and obstacle avoidance. Their proposed method is based on the Hopfield neural network along with nonlinear analog neurons which can process large amount of sensory data efficiently. Their method is capable of finding a proper path for static as well as dynamic environment where the size and the shape can be arbitrary and also the prior knowledge about the environment is unknown. The algorithm uses real-time sensing for collecting the information regarding the changes in the environment. Their approach is different from the regular neural network based approaches, since in accordance to their approach the robot can find a proper path even when the starting position and goal positions are changed continuously.

[4] Hong Qu, Simon X, Allan R. Willms and Zhang Yi [38] developed a real-time robot path planning based on a modified pulse coupled neural network model. They have taken the environment as maze type. Their approach is applicable for both static or dynamic where the complete prior information regarding the environment may or may not be known. According to their proposed neural network, neurons have lateral connection among themselves. Since the Obstacles have no connections to their neighbors. Each neuron records its parent i.e. the neighbor that caused it to fire. The real-time optimal path is then the sequence of parents from

the robot to the target. In a static case where the barriers and targets are stationary, they have proven that the generated wave in the network spreads outward with travel times proportional to the linking strength among neurons. Thus, the generated path is always the global shortest path from the robot to the target. Since the pulse propagation speed of the neurons are constant. That's why propagation of the pulse is proportional to the distance between corresponding neurons. As a result, computational complexity is solely depend on the length of the shortest path irrespective of the number of possible path exist and other kind of complexities.

3.4 Artificial Potential Field based approaches

According to Artificial potential field (APF), a point robot in C-space moves under the influence of an APF in which obstacles are assumed to generate repulsive forces and the target is assumed to generate attractive forces. The robot moves as per the resultant of these forces. This approach is known for its mathematical elegance and simplicity as path is found with very little computation [5]. But it has some disadvantages also, these are: 1) Trap situations due to local minima, 2) No passage between closely spaced obstacles, 3)Oscillations in the presence of obstacles and Oscillations in narrow passages [45]. Some APF based are described below.

[1] Guanghui Li, Atsusji Yamashita, Hajime Asama and Yusuke Tamura [43] developed An Efficient Improved Artificial Potential Field Based Regression Search Method for Robot Path Planning. They have modified the regular potential function in order to eliminate non-reachable and local minima problems, and utilize virtual local target for robot to escape oscillations and also have investigated that improvement on at potential function is not enough, since they cannot able to generate optimal or near-optimal path. That's they have introduced regression search along with the improved potential field to deal with aforementioned problem. They have shown experimentally that their proposed approach can able obtain a global optimal/near-optimal path without local minima and oscillations in complete known environment information.

[2] S. S. Ge and Y. J. Cui [44] proposed a new potential method for dynamic motion planning for mobile robot where the information regarding the environment is not known in advance. In their proposed approach, they have defined the attractive potential as a function of the relative

position and velocity of the target with respect to the robot and the repulsive potential as the relative position and velocity of the robot with respect to the obstacles and finally the virtual force as the negative gradient of the potential in terms of both position and velocity rather than position. The addition of all these improvements have made their proposed approach capable of finding a proper path in a dynamic environment.

[3] Hossein Adeli, M. H. N. Tarbrizi, Alborz Mazloomian, Ehsan Hajipour and Mehran Jahed [45] developed an iterative artificial potential field for the path planning of mobile robot. In this paper, they have modified the potential function which is iterative in nature. For this purpose, the workspace has been discretized into a grid of rectangular cells where each cell is marked as an obstacle or a non-obstacle where the potential functions for each cell is based on its distances from the destination, start and obstacles. These values are used to find the optimum points along the entire path iteratively until there are enough points for a path to be determined as a consecutive sequence of these points beginning from the start location and ending at the destination where the number of iterations depends on the size and shape of the workspace. It have been found experimentally that their approach has succeed to overcome the limitations of regular potential field approaches.

3.5 Multi-Robot Path Planning

Multi-Robot path planning is a bit complex and time consuming than single robot because it deals with multiple robot where it introduces an extra computational effort due to collision avoidance between the robots themselves along with collision avoidance between a robot and an obstacle. There exist basically two types of multi-robot path planning. These are centralised and decoupled. Centralised approaches treat the separate robots as one composite system, and typically perform the planning in a composite configuration space, formed by combining the configuration spaces of the individual robots. On the other hand, Decoupled approaches first generate path for the separate robots more or less independently and then consider the interactions between the robots [40]. Some approaches of multi-robot are described below.

[1] Fedor A. Kolushev and Alexender A. Bogdanov [39] proposed a multi-agent optimal path planning for mobile robot in environment with obstacles in real-time. They have used graph representation of the environment where all the robots are considered as dynamic obstacles. Each edges of the graph has two weights. These are distance and motion time i.e. speed which can be modified during path planning. In addition to that, expert rules for speed and path correction are synthesized to provide collision avoidance. At the end they have used Dijakstra's algorithm for finding the shortest path.

[2] Peter Svestka and Mark H. Overmars [40] in their work i.e. coordinated path planning for multiple robots has investigated that since their approach is based on coordinated planning. That's why their proposed approach is probabilistically complete i.e. their approach can solve any solvable problem with in finite amount of time. Their approach is only applicable for static environment having prior knowledge about the environment in advance. They have used a specific data structure to store multi-robot motion. It has been constructed via two steps. In first step, a roadmap is constructed for just one robot with the help of the probabilistic path planner, which can be easily applied to different robot types. In the second step, a number of these simple roadmaps are combined into a roadmap for the composite robot. They have applied their proposed algorithm on car-like robot and also have shown experimentally that their approach is capable of finding proper path even in complex environment in the order of seconds, after a preprocessing step that consumes, at most, a few minutes.

[3] Wang Mei and Wu Tie-jun [41] developed a cooperative co-evolution based distributed path planning of multiple robots in order to keep track of the movement of multiple robots in 2D world. They have taken different population for different robot where the fitness of each individual is also depend on the cooperation of the individuals of other robots. In order to keep the robot away from the obstacle, the distance between the robots and obstacles also have been added to the fitness. The proposed method is executed asynchronously and in parallel for every robot. At the end, the optimal path of each robot will be the result of co-evolution in all populations.

[4] Hong Qu, Ke Xing and Takacs Alexander [42] developed an improved genetic algorithm with co-evolutionary algorithm for global path planning of multiple mobile robot. Their approach is applicable for static environment. Every robot have separate population of chromosome. In accordance to their approach, if there is n number of robot then n numbers of GAs will be executed parallely on their corresponding population and at the end of each iteration or generation there is an information interaction step among all the GAs which is based on the island model. This step is responsible for finding out whether the robots are colliding with themselves or not.

3.6 Some Other Approaches

There exist a number of other approaches for solving robot path planning problem. Like Simulated Annealing, Dynamic programming based approach [46], fuzzy logic [47], A^* [50], cell decomposition [49] etc. Some of the approaches are demonstrated below.

[1] Allan R. Willms and Simon X. Yang [46] presented Dynamic programming for collision free robot path planning problem where the environment is real time in nature which indicates that their approach is applicable for a situation where the targets and barriers are allowed to move without any prior information about their movement. For this purpose they have taken topologically organized map as their environment where each grid point on the map has only local connections to its neighbor grid points from where it receives information in real time. Their approach is also applicable for the scenario when the barriers are static. They have shown experimentally that dynamic system converges in a small number of iteration to a state where the minimal distance to a target is recorded at each grid point and their approach have always succeeded to choose an optimal path.

[2] Meng Wang and James N. K. Liu [47] proposed fuzzy logic based robot path planning in unknown environment. They have used memory grids and there are two types of grids. One is the obstacle memory dots which represents the fuzzy possibility regarding uncertainty of the obstacles detected by sonar sensor and the other one is the trajectory memory dots to keep track trajectories traversed by the robot. They have proven theoretically that their proposed approach is

capable of finding global optimal path even in the long-wall, unstructured, cluttered, maze-like, and modified environments.

[3] Hui Miao and Yu-Chu Tian [48] investigated in their work i.e. Robot Path Planning in Dynamic Environments Using a Simulated Annealing Based Approach that the simulated annealing is efficient for path planning in an environment having different shaped obstacles, specially sharp obstacles and also convergence rate is fast enough than most of the approaches even in complex environment. Their proposed approach is capable of finding optimal or near optimal solution for both static and dynamic environment. They have taken both the moving and static obstacles as bounding polygon whose vertices form the search spaces. A mathematical model has also been developed for finding out the possibility of collision between a moving obstacles and the robot.

Chapter 4: Genetic Algorithms

4.1 Introduction to Genetic Algorithms

Genetic Algorithms (GAs) are generally portrayed as search procedure which can optimize using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. They perform a multi-dimensional search in order to provide an optimal value of an evaluation function, also known as fitness function in an optimization problem. Unlike conventional search methods, GAs deal with multiple solutions simultaneously and compute the fitness function values for these solutions. GAs are theoretically and empirically found to provide global near-optimal solutions for various complex optimization problems in the field of operation research, VLSI design, Pattern Recognition, Robot Path Planning, Bioinformatics, Game Theory, Computer-automated design, Mechanical Engineering, Climatology, Power Electronics, Economics etc. [51, 52, 53].

4.2 Background of Genetic Algorithms:

John Holland firstly put forward GAs in 1960s [54], when he worked on the studies of cellular automata with his colleagues and his students at the University of Michigan. GA became popular through his book Adaptation in Natural and Artificial Systems (Holland, 1975). The research of GA was limited to the theoretical part until the First International Conference on Genetic Algorithm, which was held in Pittsburgh, Pennsylvania in the mid-1980s. As the development of computer programs and the demand of practical application grew, GA becomes more popular within practical application.

In 1989, Evolver (Markoff, 1990-08-29) which was written by John Markoff, The New York Times' technology writer, described the application of GA to business for the first time. After that, GA has been developed rapidly. Most of Fortune 500 companies apply GA to making a time list, data analysis, the future trend forecast, budget, and solving other combinatorial optimization problems.

4.3 Complete Description of Genetic Algorithms

In GAs every solutions are encoded as chromosomes and some predefined number of chromosomes constitute the population. Genetic operators are applied on the individuals of the population on every iteration / generation until some termination criteria is fulfilled. The genetic operators are Selection, Crossover and Mutation. Elitist strategy is also applied to GAs to copy the best chromosome of the previous generations into the current generations.

To make comparison among the chromosomes of a population, fitness function is used in GAs. Fitness function evaluates how good a chromosome is relative to other chromosomes in the population. This fitness function may be a minimization or maximization function depending upon the problem scenario. The complete flowchart of Genetic algorithm has shown in Figure 6.

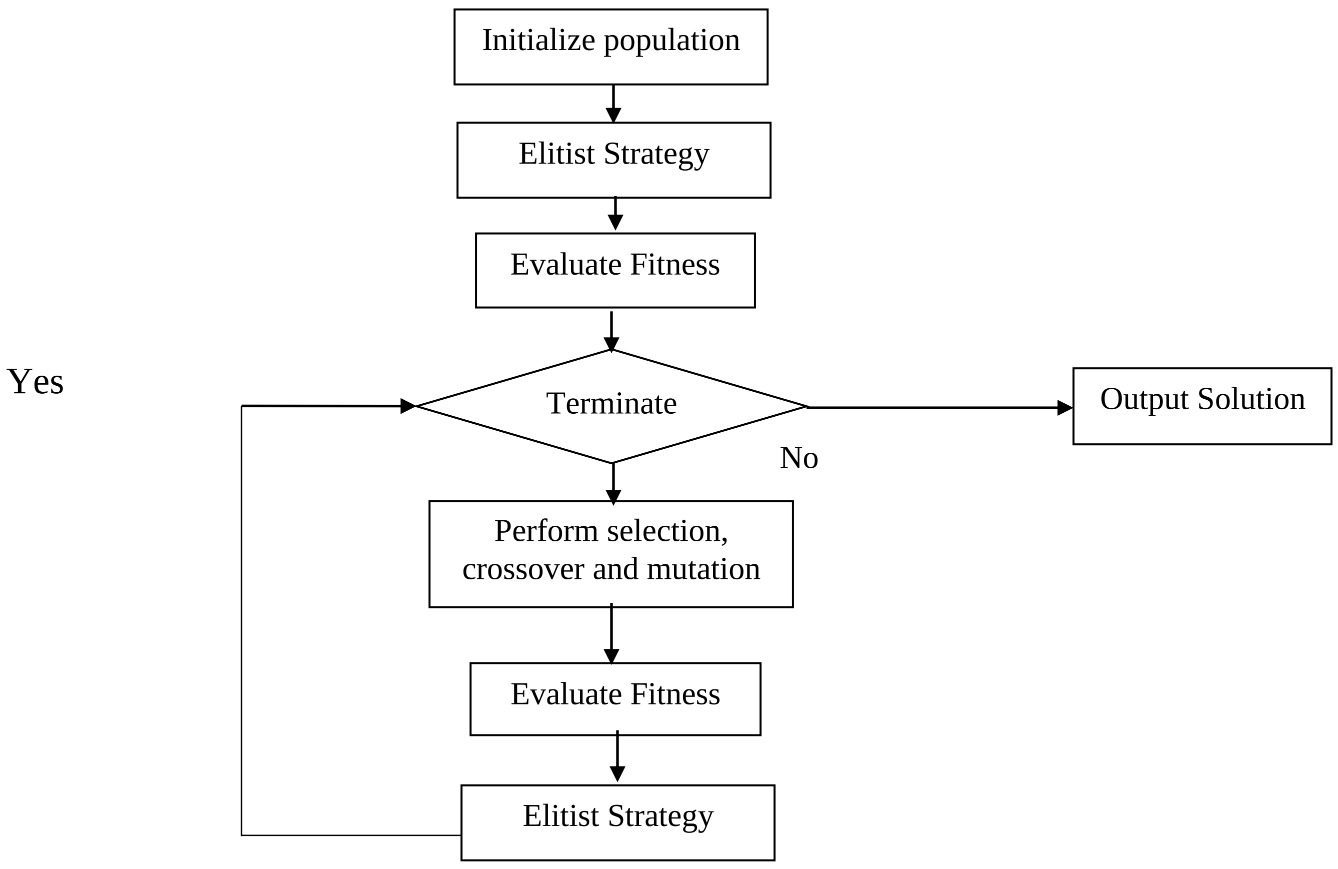


Figure 6: Flowchart of Genetic Algorithms

4.3.1 Chromosome

Every individuals or solutions in GAs are considered as chromosome.

4.3.2 Gene

Chromosome is a sequence of genes. So, the basic building block of chromosomes are Gene.

Gene 1	Gene 2	Gene 3	………	Gene n
$\longleftarrow$ Chromosome $\longrightarrow$				

Example of Chromosome have n no of genes

4.3.3 Chromosome Encoding

Chromosome encoding is one of the important task in GAs. It is the process of representing genes thereby representing chromosome also. It is way of encoding a problem solution into the solution of GAs i.e. chromosome. There exists various ways of encoding a chromosome [55]. Some of them are described below.

4.3.3.1 Binary Encoding

Binary encoding is one of the most common encoding techniques where each chromosome is composed of a string of binary bits i.e. 0 or 1. Each bit in the string can represent some characteristics of the solution. Every bit string therefore is a solution but not necessarily the best solution. The following represents a binary encoded chromosome.

Chromosome	10100010010101010101011

Example of binary encoding

4.3.3.2 Real Number Encoding

In Real Number Encoding, the chromosomes are encoded as a string of real number. Sometime there exist some range on the numbers that can be used for chromosome encoding. It is also some time called permutation encoding. The following is an example of Real Number Encoding of chromosome.

Chromosome	1 2 3 4 7 6 9 8

Example of real number encoding

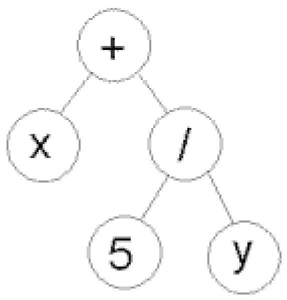
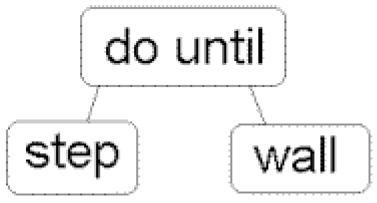
4.3.3.3 Value Encoding

In this type of encoding the chromosomes are the string of values where the values can be anything related to the problem. This type of encoding generally used for some complicated type of problems where the Binary Encoding may not be applicable or very difficult to implement. The crossover and mutation operators used on this value encoded chromosomes are different from regular crossover and mutation operator. The following is a collection of examples of value encoded chromosome.

Chromosome 1	12.3 45.6 22.6 12.1 33.2 18.9 53.9
Chromosome 2	A B H J S O P
Chromosome 3	Right Left Up Left Down Right UP

Example of value encoding

4.3.3.4 Tree Encoding

Chromosome A	Chromosome B
	
(+ x (/ 5 y))	(do_until step wall)

Example of tree encoding

In Tree Encoding, every chromosome is a tree of some objects such as functions and commands of a programming language. This encoding is mainly used for evolving program expressions for genetic programming. Some Tree Encoding based chromosome are shown below [72].

4.3.4 Population

A predefined number of chromosomes or individuals constitute the population. Initially the chromosomes are chosen using some techniques or randomly to form the population which is called initial population. Choosing a proper size of the population or in other words, how many chromosomes will be there in a population is one of the difficult as well as important task for the GAs. Generally, there do not exist any guideline for choosing appropriate population size [53]. As the population size increases, the searching process becomes more diverse but it introduces extra computation cost. So, it is very important to choose the population size in such a way to make a balance between the diversity in the searching process and the computational cost. Note that it has been shown in [53, 56] that as the number of generations or iterations goes to infinity the elitist model of GAs will able to converge into optimal solution for any population size.

4.3.5 Fitness

The purpose of fitness function is to make comparison among the chromosomes of a population in GAs. Fitness function is also sometimes called objected function. This fitness function may be a minimization or maximization function depending upon the problem scenario.

It is very difficult to calculate fitness function in the case of multi criterion or multi objective optimization. A confusion arises regarding how to determine if one solution is better than another. Because, it may happen that one chromosome is better than another chromosome with respect to one criterion but worse with respect to another. One solution to this problem is to use the combination of different criteria as the fitness function. But, for more advanced problems, it may be useful to consider something like Pareto optimally or others ideas from multi criteria optimization theory [55].

4.3.6 Selection

Selection operator is based on the concept of "survival of the fittest" or in other words, the probability of selection of a particular chromosome is proportional to the fitness value of that chromosome in hopes that their off springs may have higher fitness.

There are mainly two types of selection scheme, proportionate selection and ordinal-based selection. Proportionate-based selection picks out individuals based upon their fitness values relative to the fitness of the other individuals in the population. Ordinal-based selection schemes selects individuals not upon their raw fitness, but upon their rank within the population. This requires that the selection pressure is independent of the fitness distribution of the population, and is solely based upon the relative ordering (ranking) of the population [55]. The selected chromosomes form a mating pool whose size are generally taken as the same as that of the size of the population. After this step is over, the crossover and mutation operator are applied on the chromosomes of the mating pool to generate a new population. There exists various types of selection strategies [55, 57]. Few of them are described below.

4.3.6.1 Roulette Wheel Selection

As the name suggest this selection techniques is inspired from Roulette wheel in casino which includes 37 colored and numbered pockets on the wheel and a small marble will be thrown to choose a number randomly. Here in this case, the number of pockets is equal to the number of chromosomes in the mating pool and the size of the pockets are proportional to the fitness of the corresponding chromosome assigned to it. As a result, the probability of the selection of chromosome from the mating pool is proportional to its fitness value. That's why Roulette Wheel Selection is also called Fitness Proportionate Selection. The principle of roulette selection is a linear search through a roulette wheel with the slots in the wheel weighted in proportion to the individual's fitness values. A target value is set, which is a random proportion of the sum of the fit nesses in the population. The population is stepped through until the target value is reached. This is only a moderately strong selection technique, since fit individuals are not guaranteed to be selected for, but somewhat have a greater chance. A fit individual will contribute more to the target value, but if it does not exceed it. Then the next chromosome in line has a chance, and it may be weak. It is essential that the population not be sorted by fitness, since this would dramatically bias the selection. Sometimes, binary search having time complexity $O(\log n)$ can be used in place of linear search having time complexity $O(n)$.

The algorithm of the Roulette Wheel Selection is shown is Algorithm 1. According to the algorithm the expected value of an individual is that fitness divided by the total fitness of the population. Each individual is assigned a slice of the roulette wheel, the size of the slice being proportional to the individual's fitness. The wheel is spun N times, where N is the number of individuals in the population. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next generation

```
1. $n =$ the total number of chromosome in the population.
2. $f[] =$ fitness values of n chromosomes.
3. $p[] =$ mating pool.
4. $p_{new} = []$.
// it will contain n number of selected chromosome.
5. $e = []$
// it will contain the expected fitness values of n chromosomes.
6. $f_{total} = 0$.
// it will contain the total summation of the fitness values of n chromosomes.
7. for $i = 1$ to $n$:
        a. $f_{total} = f_{total} + f[i]$
8. for $i = 1$ to $n$:
        a. $e[i] = f[i]/f_{total}$
9. $sum = 0$.
// it will contain the summation of fitness values during linear search.
10. for $i = 1$ to $n$:
        a. $rand =$ generate a random number in the range $[0, 1)$
        b. $j = 1$.
        c. while $sum < rand$ :
                i. $sum = sum + e[j]$.
                ii. $j = j + 1$.
        d. $p_{new[i]} = p[j]$.
11. return $p_{new}$.
```

Algorithm 1: Algorithm for roulette wheel selection

Since in roulette wheel selection the chromosomes having higher fitness values are selected again and again most of the time. That why the convergence rate of this selection scheme is higher but sometimes, it leads to premature convergence.

4.3.6.2 Rank Based Selection

```
1. $n =$ the total number of chromosome in the population.
2. $f[] =$ fitness values of n chromosomes.
3. $p[] =$ mating pool.
4. $p_{new} = []$
// it will contain the n number of selected chromosome.
5. $e = []$
// it will contain the selection probabilities of n chromosomes after mapping.
6. $s[] =$ the chromosomes according to their fitness values.
7. Sort the chromosomes in the mating pool $p[]$ according to their fitness values $f[]$ and store them in $s[]$.
8. Assign the new selection probabilities of each chromosome using the mapping function and store them in $e[]$.
9. $sum = 0$.
// it will contain the summation of fitness values during linear search.
10. for $i = 1$ to $n$:
        a. $rand =$ generate a random number in the range $[0, 1)$
        b. $j = 1$.
        c. while $sum < rand$:
                i. $sum = sum + e[j]$.
                ii. $j = j + 1$.
        d. $p_{new[i]} = s[j]$.
11. return $p_{new}$.
```

Algorithm 2: Algorithm for rank based selection

If any one of the chromosome in the mating pool having fitness value too high than the rest of the chromosomes and occupies around 80% of the area in the roulette wheel. Then it is obvious that during selection most of the times the aforementioned chromosome will be selected and as a consequence other chromosomes have too few chances to be selected which leads to premature convergence. To overcome this problem Rank Based Selection techniques is used.

Rank-based selection is the selection strategy where the probability of a chromosome being selected is based on its fitness rank relative to the entire population. Rank-based selection schemes first sort individuals in the population according to their fitness and then computes selection probabilities according to their ranks rather than the fitness values. Hence rank-based selection can maintain a constant pressure in the evolutionary search where it introduces a uniform scaling across the population and is not influenced by super-individuals or the spreading of fitness values at all as in proportional selection. Rank-based selection uses a function to map the indices of individuals in the sorted list to their selection probabilities. Although this mapping function can be linear (linear ranking) or non-linear (non-linear ranking), the idea of rank-based selection remains unchanged. The performance of the selection scheme depends greatly on this mapping function.

In Rank Based Selection though the convergence rate is slower than Roulette Wheel Based Selection but better for avoiding premature convergence. But this selection scheme can be computationally expensive because of the need to sort populations [57]. The algorithm for Rank Based Selection is shown in algorithm 2.

4.3.6.3 Tournament Selection

```
1. $n =$ the total number of chromosomes.
2. $f[] =$ fitness values of n chromosomes.
3. $p[] =$ population.
4. $t_{size} =$ Tournament Size
5. $p_{new} = []$.
// it will contain the selected n chromosomes.
6. for $i = 1$ to $n$:
        a. $temp = []$.
        // it will contain $t_{size}$ no of chromosomes.
        b. Select $t_{size}$ no of chromosome randomly from $p[]$ and assign them into $temp$.
        c. $best = []$.
        d. Find out the best chromosome in $temp[]$ and assign it to $best[]$.
        e. $p_{new[i]} = best$.
7. return $p\_new$.
```

Algorithm 3: Algorithm for tournament selection

Tournament selection is one of the popular selection techniques. Tournament selection involves running several tournaments from there a few individuals chosen at random from the population. They complete among themselves and the chromosome having best fitness value wins the tournament. Then the winner chromosome is sent to mating pool and the process is repeated until the mating pool for generating new offspring is filled. In Tournament Selection the selection pressure is easily adjusted by changing the tournament size which is generally taken as two. Tournament selection gives a chance to all individuals to be selected and thus it preserves diversity, although keeping diversity may degrade the convergence speed. But the diversity is solely depend on the tournament size. Because, if the tournament size is larger, weak individuals have a smaller chance to be selected. The selective pressure can also be updated by the difference in fitness values of the chromosomes in the mating pool after the selection. Tournament selection has several benefits over alternative selection methods for genetic algorithms (for example, roulette wheel selection and rank based selection) it is efficient to code, allows the selection pressure to be easily adjusted. Tournament selection is also efficient in terms of computational complexities since it can be implemented parallely and it do not include any sorting mechanism. The Algorithm for Tournament Selection is described in algorithm 3.

4.3.7 Crossover

Crossover operator is generally applied on two randomly selected chromosomes from the mating pool which are known as parent chromosome where they exchange information between themselves and generate two children for the next generation. The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Basically crossover helps to converge towards a local optima.

Some probability is used in GAs to determine whether the Crossover will take place or not. This is called Crossover probability. The values of the crossover probability are predefined depending upon the problem scenario on which it is being applied and can be changed adaptively in every iteration or generation. There are various types of crossover techniques used in GAs [55]. Some of them are explained below.

4.3.7.1 Single Point crossover

Single Point Crossover is the most widely used crossover techniques, used in Genetic Algorithms. After selecting two parent chromosomes form the mating pool, at first a crossover position is selected randomly. Then rest of the part of the parent chromosomes (i.e. the part after the randomly selected position) are exchanged between themselves and generate two offspring for the next generation. As an example of single point crossover as shown below, consider a pair of chromosome ch1 and ch2 of size n each.

Ch1	$\alpha_1, \alpha_2, \dots \dots \dots \dots \dots \dots, \alpha_{n-1}, \alpha_n$
Ch2	$\beta_1, \beta_2, \dots \dots \dots \dots \dots \dots, \beta_{n-1}, \beta_n$

Child1	$\alpha_1, \alpha_2, \dots \dots, \alpha_{pos}, \beta_{pos+1}, \dots \dots, \beta_{n-1}, \beta_n$
Child2	$\beta_1, \beta_2, \dots \dots \dots, \beta_{pos}, \alpha_{pos+1} \dots \dots \dots, \alpha_{n-1}, \alpha_n$

Suppose a crossover position (pos) is selected randomly from the range $[1, n-1]$. So after performing crossover at the position(pos) , a pair of children child1 and child2 are generated.

4.3.7.2 Two Point crossover

In two-point crossover at first two crossover positions are chosen. Then contents between these positions are exchanged between two mated parents. It should be noted that adding further crossover points reduces the performance of the GA. However, an advantage of having more crossover points is that the problem space may be searched more thoroughly [55]. As shown below, consider a pair of chromosome ch1 and ch2 of size n each. Suppose two crossover positions ($pos1$ and $pos2$) are selected. Then during crossover, the contents between $pos1$ and $pos2$ are exchanged between ch1 and ch2 and as a consequence a pair of children child1 and child2 are generated as shown below.

Ch1	$\alpha_1, \alpha_2, \dots \dots \dots \dots \dots \dots, \alpha_{n-1}, \alpha_n$
Ch2	$\beta_1, \beta_2, \dots \dots \dots \dots \dots \dots, \beta_{n-1}, \beta_n$

Child1	$\alpha_1, \alpha_2, \dots \dots, \beta_{pos1}, \dots, \beta_{pos2}, \dots \dots, \alpha_{n-1}, \alpha_n$
Child2	$\beta_1, \beta_2, \dots \dots \dots, \alpha_{pos1}, \dots, \alpha_{pos2} \dots \dots \dots, \beta_{n-1}, \beta_n$

Example of two point crossover

4.3.7.3 Multi Point crossover

Multi Point Crossover also known as N-Point crossover, can be done in two ways i.e. either by even number of crossover positions or by odd number of crossover positions. In the case of even number of crossover, crossover positions are selected randomly around a circle and information is exchanged. In the case of odd number, a different crossover position is always assumed at the beginning of the chromosome.

Originally, GAs were using one-point crossover which cuts two chromosomes in one point and splices the two halves to create new ones. But with this one-point crossover, the head and the tail of one chromosome cannot be passed together to the offspring. If both the head and the tail of a chromosome contain good genetic information, none of the generated offsprings will share the two good features. 2-point crossover can be used to avoid this drawback, and that's why 2-pont crossover is generally considered better than 1-point crossover. In fact this problem can be generalized to each gene position in a chromosome. Genes that are close on a chromosome have more chance to be passed together to the offspring obtained through a *N*-points crossover. It leads to an unwanted correlation between genes next to each other. Consequently, the efficiency of a N-point crossover will depend on the position of the genes within the chromosome. In a genetic representation, genes that encode dependent characteristics of the solution should be close together. To avoid all the problem of genes locus, a good thing is to use a uniform crossover which has been described next [55].

4.3.7.4 Uniform crossover

The implementation of uniform crossover is a bit different for differently encoded chromosome. In case of binary encode chromosome, each gene in the offspring is created by copying the corresponding gene from one or the other parent, chosen according to a random generated binary crossover mask which has the same length as that of the chromosomes. Where if there is a 1 in the crossover mask, the gene is copied from the first parent, and in case of a 0 in the mask the gene is copied from the second parent. For another child the rule is just opposite. Consider the following example as shown below, It can be noticed, that while producing child1, when there is a 1 in the mask, the gene is copied from the ch1 else from the ch2. On producing child2, when there is a 1 in the mask, the gene is copied from ch2, when there is a 0 in the mask; the gene is copied from the ch1.

Ch1	0 1 0 0 0 1 1 0 1 0 1 0 1
Ch2	1 0 0 1 1 0 1 0 1 1 0 1 0
Mask	0 1 0 1 0 1 0 1 0 1 0 0 0
Child1	1 1 0 0 1 1 1 0 1 1 0 1 0
Child2	0 0 0 1 0 0 1 0 1 1 1 0 1

Example of uniform crossover for binary encoded chromosome

In case of Real Number Encoding also a crossover mask is used in order to understand from which parent chromosome the gene has to be selected. The crossover mask has a length that is equal to the length of the parent chromosome where each and every element is a random number, taken from the range [0, 1) and a mask-probability ($Mask_P$) is also used. In case of first child, for a gene belongs to the first parent chromosome having mask value less than $Mask_P$ will be selected from the first parent otherwise from the second parent. The rule for the genes of second child is just the reverse. As shown in the following example, in case of 1st child if a gene having mask value less than 0.50 which is taken as $Mask_P$, will be selected from ch1 otherwise from the ch2. In case of 2nd child a gene will be selected from ch1 if it has mask value greater than $Mask_P$, otherwise from the ch2.

Ch1	10.25 3.36 8.89 5.34 9.89 3.90
Ch2	4.56 7.89 3.21 7.65 9.78 1.12
Mask	0.23 0.45 0.87 0.57 0.98 0.21
Mask Probability	0.50
Child1	10.25 3.36 3.21 7.65 9.78 3.90

Child2	4.56 7.89 8.89 5.34 9.89 1.12

Example of Uniform Crossover for real value encoded chromosomes

The offsprings or children generated due to uniform crossover are therefore contain a mixture of genes from each parent. The number of effective crossing point is not fixed which is on the average $n/2$ (where n is the chromosome length) [55].

4.3.7.5 Three Parent Crossover

Three Parent Crossover is generally applicable for binary encoded chromosomes. Unlike all the aforementioned crossover techniques, three parent crossover needs three parent chromosomes for performing crossover and generates single child. In Three Parent Crossover, Each bit of the first parent is compared with the bit of the second parent. If both are the same, the bit is taken for the offspring otherwise; the bit from the third parent is taken for the offspring. An example of Three Parent Crossover has been shown below.

Ch1	1 0 0 1 0 1 0 1 0 0 1 1
Ch2	1 1 0 1 0 1 0 1 0 0 0 1
Ch3	0 1 0 1 0 1 0 0 0 1 0 0
Child	1 1 0 1 0 1 0 1 0 0 0 1

Example of three parent crossover

4.3.7.6 Shuffle crossover

Shuffle crossover is related to uniform crossover. Most commonly Shuffle Crossover is implemented with Single Point Crossover. But before the genes are exchanged, they are randomly shuffled in both parents. After recombination, the genes in the offspring are unshuffled [55, 58]. An example is shown below.

Ch1	1 0 0 1 0 0 1 0 0 1
Ch2	0 1 0 1 0 0 1 0 1 0

Shuffle: (1 to 3, 3 to 7, 7 to 9, 9 to 10, 10 to 1)

After Shuffling

Ch1	1 0 1 1 0 0 0 0 1 0
Ch2	0 1 0 1 0 0 0 0 1 1

Performing single point crossover at position 4 (randomly chosen)

Child1	1 0 1 1 0 0 0 0 1 1
Child2	0 1 0 1 0 0 0 0 1 0

After unshuffling

Child1	1 0 0 1 0 0 1 0 1 1
Child2	1 1 0 1 0 0 1 0 0 0

Example of shuffle crossover

4.3.7.7 Crossover with Reduced Surrogate

The reduced surrogate operator constrains crossover to always produce new individuals wherever possible. This is implemented by restricting the location of crossover points such that crossover points only occur where gene values differ [59].

4.3.7.8 Arithmetic Crossover

Arithmetic crossover is applicable for real number encoding. Arithmetic crossover operator linearly combines the two parent chromosomes [55]. After the selection of two parent chromosomes ch1 and ch2 from the mating pool, the generated offsprings or the children child1 and child2 are shown below. The weight a has been used in the equations 4.1 of child1 and 4.3 of child2 is the weight which governs dominant individual in reproduction and it is between 0 and 1 [60].

$$\text{Child1} = a * ch1 + (1 - a) * ch2 \ldots\ldots\ldots\ldots\ldots\ldots.4.1$$

$$\text{Child2} = a * ch2 + (1 - a) * ch1 \ldots\ldots\ldots\ldots\ldots\ldots..4.2$$

4.3.7.9 Ordered Crossover

Ordered Crossover was proposed by Davis and also used for chromosomes with permutation encoding [61]. After selecting two parent chromosomes ch1 and ch2 from the mating pool, two random crossover points are selected partitioning them into a left, middle and right portion. The ordered two-point crossover behaves in the following way: child1 inherits its left and right section from ch1, and its middle section is determined by the genes in the middle section of ch1 in the order in which the genes are appeared in ch2. A similar process is applied to determine child2. An example of ordered crossover is shown below.

Parent Chromosome	Left	Middle	right
Ch1	3 2	8 1	7 5
Ch2	2 1	7 2	8 9

Children	Left	Middle	right
Child1	3 2	1 8	7 5
Child2	2 1	2 7	8 9

4.3.7.10 Partially Mapped Crossover (PMX)

Partially Mapped Crossover also known as Partially Matched Crossover was proposed by [62]. PMX can be applied in the TSP problem where chromosomes are simply sequences of integers, where each integer represents a different city and the order represents the time at which a city is visited. According to this crossover technique, after selecting two parent chromosomes from the mating pool, two crossover positions are randomly chosen. The region between these two crossover positions is called crossover region or matched section which will be exchanged between two parent chromosomes. But after performing crossover some duplication of genes may arise due to the crossover region where cross-referencing with the parent of the alternate chromosome is used for that purpose.

Ch1	3 2 1	6 5 4	9 8 7
Ch2	8 4 5	1 7 9	2 3 6
Child1	3 2 1	1 7 9	9 8 7
Child2	8 45	6 5 4	2 3 6

Consider the following example where ch1 and ch2 are the parent chromosome and the crossover positions are 3 and 7 which have been taken randomly. Two children child1 and child2 are generated after performing crossover which have some duplicated genes due to the crossover region. A number of relations can be found from the crossover regions which are $6 \leftrightarrow 1, 5 \leftrightarrow 7\ and\ 4 \leftrightarrow 9$. These relations can be used to overcome the duplication of genes like for the relation $6 \leftrightarrow 1$, the gene 1 in child1 will be replaced by the gene 6 and the gene 6 in child2 by the gene 1. After applying all the relations the resulted children are shown below.

Child1	3 2 6	1 7 9	4 8 5
Child2	8 9 7	6 5 4	2 3 1

4.3.7.11 Cycle Crossover

Cycle crossover is applicable for permutation encoded chromosome. The Cycle Crossover operator identifies a number of so-called cycles between two parent chromosomes. To construct a cycle of genes from parent1, it is needed to start with the first gene of parent1. Then look at the gene at the equal position in parent2 and go to the position with the same gene in Parent1, insert this gene to the cycle and repeat above mentioned step until the destination i.e. the first gene of parent1 is being arrived. After creation of the first cycle, if any gene in the parent chromosome left untraversed or have not been used in the first cycle then the next cycle will be started from the first gene of parent1 have not been used in first cycle. The process will continue until all the genes of the parent chromosome will be traversed. At the end, to form Child 1, cycle 1 is copied from parent 1, cycle 2 from parent 2, cycle 3 from parent 1, and so on according to their respective positions in parent chromosomes. An example of Cycle Crossover has been shown below.

Ch1	1 7 5 3 2 8 0 4 9 6
Ch2	8 3 6 4 5 1 2 0 9 7

Cycle 1: Cycle 1 will be started from the first gene of Parent 1 (ch1) i.e. 1 and the gene in the first position of Parent 2 (ch2) is 8. So, 1 goes to 8. Then the position of gene 8 in ch1 is 6 and

the gene in the sixth position of ch2 in 1 which is destination of cycle 1. So, the values in Cycle 1 are 1, 8. 1 and 8 are marked blue.

Ch1	1 7 5 3 2 8 0 4 9 6
Ch2	8 3 6 4 5 1 2 0 9 7

Cycle 2: Cycle 2 will be started from the second gene of ch1 i.e. 7 and the second gene in ch2 is 3. Then 3 is at fourth position in ch1 and the fourth gene of ch2 is 4. Then 4 is at eighth position in ch1 and the eighth gene in ch2 is 0. Then 0 is at seventh position in ch1 and in ch2 the seventh gene is 2. Then 2 is at fifth position in ch1 and the gene in fifth position of ch2 is 5. After that, 5 is at third position in ch1 and the third gene in ch2 is 6. Then 6 is tenth position in ch2 and the tenth chromosome in ch2 in 7 which is the destination of cycle 2. So, the values of cycle 2 are 7, 3, 4, 0, 2, 5, 6. These have been marked green.

Ch1	17 5 3 2 80 4 9 6
Ch2	83 6 4 5 12 0 9 7

Cycle 3: Cycle 3 will be started with 9 and also ended with 9. Since the gene 9 is in the same position in both the parent chromosome.
So, finally child1 will get the gene form the ch1 for cycle 1. Then from the ch2 for cycle and at last from the ch1 for cycle 3. The rule for child2 is just the reverse of child1.

Child1	1 3 6 4 5 8 20 9 7
Child2	87532 1 04 9 6

4.3.8 Mutation

Mutation operator is applied on each and every chromosome in the mating pool after the crossover operation is over. Unlike crossover operator, it introduces some genetic diversity into the population in order to explore the searching process over the solution space and avoid to get stuck at local optima or in other words, avoid premature convergence.

Before performing mutation, a random number $rand$ has to generate from the range [0, 1). Then if the $rand \leq P_{Mut}$ then only mutation will occur where P_{Mut} is some predefined Mutation probability. In order to obtain the optimal solution, one needs to maintain the population diversity by keeping the mutation probability high. On the other hand, as the optimal solution is being approached, fewer changes in the present solutions are necessary to move in the desired direction. This implies that the mutation probability needs to be reduced as the number of iterations or generations increases. Sometimes, at any stage of the algorithm, many changes in the present best solution are required in order to get the optimal solution. Thus to have an efficient search process with GAs, the variation of the mutation probability with the iteration or generation can be made variable [53]. There are various type of mutation for various types of chromosome representations, some of them are described below.

4.3.8.1 Flipping

Flipping only applicable for binary encoded chromosome. After selecting a chromosome form the mating pool, flipping involves changing the gene value of the chromosome form 1 to 0 and from 0 to 1 [55]. An example of Flipping is shown in below, where the parent chromosome is ch1 and randomly position 3, 7 10, 12 and 13 have been selected for flipping randomly and generate ch1'.

Ch1	1 0 **1** 0 1 0 **1** 0 0 **1** 1 **10** 1 0
Ch1'	1 0 **0** 0 1 0 **0** 0 0 **1** 1 **01** 1 0

4.3.8.2 Interchanging

In interchanging, at first two positions are selected in the chromosome which have been selected for performing mutation from the mating pool. Then the gene at the selected position are interchanged between themselves [55]. Example of interchanging has been shown below where $i-th$and $j-th$ gene positions in ch1 have been selected randomly for interchanging and generates ch1'.

Ch1	$\alpha_1, \alpha_2, \dots\dots, \alpha_{i-1}, \boldsymbol{\alpha_i}, \alpha_{i+1}, \dots \ \dots, \alpha_{j-1}, \boldsymbol{\alpha_j}, \alpha_{j+1}, \dots\dots, \alpha_{n-1}, \alpha_n$
Ch1'	$\alpha_1, \alpha_2, \dots\dots, \alpha_{i-1}, \boldsymbol{\alpha_j}, \alpha_{i+1}, \dots \ \dots, \alpha_{j-1,}\boldsymbol{\alpha_i}, \alpha_{j+1}, \dots\dots, \alpha_{n-1}, \alpha_n$

4.3.8.3 Reversing

After selecting a random position in the chromosome and the genes next to that position are reversed. In the example shown below, the $i-th$ position has been selected in ch1 randomly for reversing and generates ch1'.

Ch1	$\alpha_1, \alpha_2, \dots \dots, \boldsymbol{\alpha_i}, \alpha_{i+1} \dots \ \dots \dots \dots, \alpha_{n-1}, \alpha_n$
Ch1'	$\alpha_1, \alpha_2, \dots \dots, \boldsymbol{\alpha_i}, \alpha_n, \alpha_{n-1} \ \dots \ \dots \dots \dots, \alpha_{i+1}$

4.3.8.4 Scramble

According to this techniques, at first two random positions are selected in the chromosome, selected for mutation and then scramble the genes in between the two randomly selected positions [63]. The example is shown below where position $i-th$ and $(i+3)-th$ have been selected randomly in the chromosome ch1 and then the chromosome ch1' is generated after scrambling.

Ch1	$\alpha_1, \alpha_2, \dots \dots, \boldsymbol{\alpha_i}, \alpha_{i+1}, \alpha_{i+2}, \boldsymbol{\alpha_{i+3}}, \dots \ \dots \dots \dots, \alpha_{n-1}, \alpha_n$
Ch1'	$\alpha_1, \alpha_2, \dots \dots, \alpha_{i+3}, \alpha_{i+2}, \alpha_i, \alpha_{i+1} \ \dots \ \dots \dots \dots, \alpha_{n-1}, \alpha_n$

4.3.8.5 Inversion

Inversion mutation is generally applicable for permutation representation of chromosome. According to this strategy, after the random selection of two gene position, the intermediate genes are reversed. So, it can be said that inversion is type of scramble [64] As shown in following example, $i-th$ and $(i+3)-th$ position are selected randomly and then the intermediated genes are reversed to generate the ch1'.

Ch1	$\alpha_1, \alpha_2, \dots \dots, \boldsymbol{\alpha_i}, \alpha_{i+1}, \alpha_{i+2}, \boldsymbol{\alpha_{i+3}}, \dots \ \dots \dots \dots, \alpha_{n-1}, \alpha_n$
Ch1'	$\alpha_1, \alpha_2, \dots \dots, \alpha_{i+3}, \alpha_{i+2}, \alpha_{i+1}, \alpha_i \ \dots \ \dots \dots \dots, \alpha_{n-1}, \alpha_n$

4.3.8.6 Displacement

In displacement mutation, at first two gene position are selected randomly and consider the intermediate genes as a subpart of chromosome. After that again select a gene position randomly outside that subpart of the chromosome and then reinsert the subpart into that position [64]. For example, at first two gene position are selected randomly which are $i-th$ and $(i+3)-th$ position and then the intermediate genes i.e. from $i-th$ to $(i+3)-th$ are considered as subpart of the chromosome ch1. Then again a gene position is selected randomly with outside of that subpart which is 2. As a result of that, the subpart is inserted in between the gene 1 and gene 2.

Ch1	$\alpha_1, \alpha_2, \alpha_3 \ldots \ldots, \boldsymbol{\alpha_i}, \alpha_{i+1}, \alpha_{i+2}, \boldsymbol{\alpha_{i+3}}, \ldots \ldots \ldots \ldots, \alpha_{n-1}, \alpha_n$
Ch1'	$\alpha_1, \boldsymbol{\alpha_i}, \alpha_{i+1}, \alpha_{i+2}, \boldsymbol{\alpha_{i+3}}, \alpha_2, \ldots \ldots \ldots \ldots \ldots \ldots, \alpha_{n-1}, \alpha_n$

4.3.8.7 Inverted displacement

This inverted displacement mutation is based on displacement mutation. But unlike displacement mutation, here in this inverted mutation after selecting the subpart of the chromosome by selecting the two random position, the subpart is reinserted into random position in reverse order. As depicted below, after selecting the subpart by selecting the two random positions i.e. $i-th$ and $(i+3)-th$, this selected subpart has been reinserted into the second position of the chromosome but in reverse order.

Ch1	$\alpha_1, \alpha_2, \alpha_3 \ldots \ldots, \alpha_i, \alpha_{i+1}, \alpha_{i+2}, \alpha_{i+3}, \ldots \ldots \ldots \ldots, \alpha_{n-1}, \alpha_n$
Ch1'	$\alpha_1, \alpha_{i+3}, \alpha_{i+2}, \alpha_{i+1}, \alpha_i, \alpha_2, \ldots \ldots \ldots \ldots \ldots \ldots, \alpha_{n-1}, \alpha_n$

4.3.8.8 Insertion

Insertion mutation is one most effective mutation strategy. It is based on the concept of displacement mutation. But unlike displacement mutation, here in this case only a single gene value which is selected randomly is reinserted into a position which will be also selected randomly. Consider the following figure, here ch1 is parent chromosome where $i-th$ position is selected randomly. As a result of that the gene α_i is reinserted into the third position of parent chromosome ch1 and generates ch1'.

Ch1	$\alpha_1, \alpha_2, \alpha_3, \dots \dots \dots, \alpha_i, \dots \ \dots \dots \dots, \alpha_{n-1}, \alpha_n$
Ch1'	$\alpha_1, \alpha_2, \alpha_i, \alpha_3, \dots \dots \dots \ \dots \dots \dots, \alpha_{n-1}, \alpha_n$

4.3.9 Elitist Strategy

```
1. Copy the best string (say $S_0$) of the initial population in a separate location.
2. Until the termination condition is satisfied:
    1. Obtaining a new population (say $Q_1$ ) after applying genetic operators on each iteration.
    2. Compare the worst chromosome in $Q_1$ (say $S_1$) with $S_0$ in terms of their fitness values. If $S_1$ is found to be worse than $S_0$, then replace $S_1$ by $S_0$.
    3. Find the best string in $Q_1$ (say $S_2$) and replace $S_0$ by $S_2$.
```

Algorithm 4: algorithm for Elitist Strategy

The purpose of this method is to keep the best chromosome of the previous generation into the next generation. Because, such chromosome can be lost if they are not selected to reproduce or if crossover or mutation destroys them. The algorithm is described in algorithm 4 which is based on the elitist strategy in [53].

4.3.10 Termination Condition

Usually there no such termination conditions for purpose converging into an optimal solution [53]. Though there exist some termination conditions. Some of them are described below.

1. If the number of iterations or generations exceeds some predefined iteration numbers.
2. If the GAs exceeds some predefined time duration.

3. If the fitness value of the elite chromosome will not change for some pre specified number of generations or iterations.

4.4 Parallel Genetic Algorithms (PGAs)

PGAs are not just parallel versions of sequential genetic algorithms. In fact they actually reach the ideal goal of having a parallel algorithm whose behavior is better than the sum of the separate behaviors of its component sub-algorithms. PGAs have a number of advantages over GAs. Some of them are robustness, easy customization for a new problem, and multiple-solution capabilities. These are the characteristics that led GAs and other EAs to be worth of study and use. In addition, a PGAs are usually faster, less prone to finding only sub-optimal solutions, and able to cooperating with other search techniques in parallel [65]. There exists various types of PGAs. But these are broadly divided into three types of parallel GAs [66]. They are master-slave parallel GAs, fine-grained and coarse grained GAs which are described below.

4.4.1 Master-Slave Parallel GAs

Master-Salve Parallel GAs explore the search space in the same way as that of simple GAs. As a consequence, it is easier to implement [67]. There is a single Master Processor and a number of Slave processors in the Master-Slave Parallel GAs and it uses single population. The Master processor maintains the population and perform genetic operators on them. After that it sends the fraction of population to each of the slave processors. The slave processors are responsible for evaluating the fitness and these works are done in parallel. A figure of mater-salve parallel is in fig 7 where there is communication links between each of the slave processors and the master processors whereas no communication between the slave processors. Since the evaluation of fitness which is taken place on each of slave processors are independent of each other. That’s why there is no need of communication between the slave processors. Communication is only occurs at the time when the master processor sends the fraction of the population to the each of the slave processors for evaluating fitness and the time when the slave processors return the result at the end of their fitness evaluation to the master processor. So, the execution time of master-slave parallel GAs consist of the computation time and the communication time.

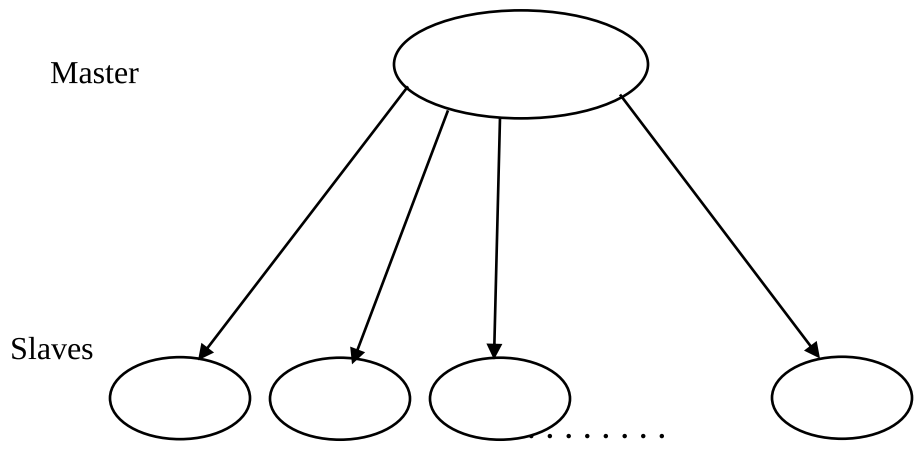


Figure 7: Image of master-slave Parallel GAs

There are generally two types of Master-Slave parallel GAs. These are synchronous and asynchronous. When the algorithm stops and waits to receive the fitness values for all the population before proceeding into the next generation, then the algorithm is synchronous and when the algorithm does not stop to wait for any slow processors, then the algorithm is known as asynchronous [66].

4.4.2 Fine Grained Parallel GAs

Fine-grained parallel GAs are useful for massively parallel computers and consist of one 2D spatially-structured processors where there is each individual for each of the processor as shown in Figure 8. Like master-slave parallel GAs, fine grained parallel GAs are also based on singe population. Sometimes this fine grained parallel GAs are also called cellular because of its similarity with cellular automata with stochastic transition rules. Fitness evaluation is done simultaneously for all individuals where selection, reproduction and mating takes place locally i.e. among the neighborhoods only. This fine grained parallel GAs can be also implemented in 1D where there will be a small number of processor on either side of the central processor [55].

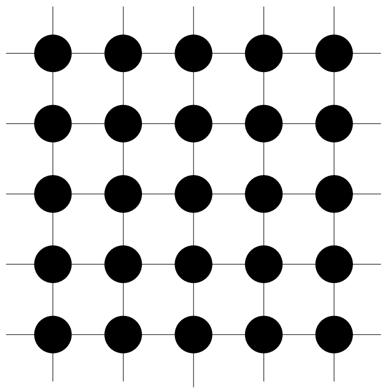


Figure 8: Image of fine grained parallel GAs

4.4.3 Corse grained Parallel GAs

Corse grained parallel GAs are also known as Distributed GAs, since they are useful for "multiple input multiple data" computers. Sometimes they are also known as island parallel GAs, since the working principle of corse grained parallel GAs are almost similar as that of "island model". Unlike master-slave parallel GAs and fine grained parallel GAs, here the population is divided into the processors. Each processors are responsible for performing genetic operators and evaluating fitness on their corresponding sub-population. In addition to that, some individuals are exchanged between the subpopulations which is known as migration. Migration can be controlled by various parameters like migration rate, topology, migration scheme like best/worst/random individuals to migrate and the frequency of migrations. The main reason for this approach is to periodically reintroduce diversity otherwise converging subpopulations. It is considered to some extent, different subpopulations will tend to explore different portions of the search space [55]. Along with that, the convergence rate of this scheme is also faster than simple GAs, due to the division of individuals among the processors. Corse grained parallel GAs can be of various types depending upon the migration techniques use on them. There exist various migration techniques. Some of them are ring topology, mesh topology, random graph, complete net topology and many others.

Chapter 5: The Proposed Method Using GAs for Single Robot Path Planning

5.1 Problem Statement

In this work an improved genetic algorithm has been developed for robot path planning problem, where some domain knowledge based operator have also been used along with the regular GAs. Here each path is associated with a start node, target and a variable number of intermediate node in between them. The aim is to get an optimal or near optimal path considering distance traversed by the robot being the optimization criteria. The environment is represented by a collection of orderly numbered grid which are rectangular shaped as shown in Figure 9. The colored grids represent the obstacles whereas the rest of the grids i.e. the blank grids represent the free spaces through which a robot can move freely. If every segments of a path are passed through the blank grid, then only the path will be known as a feasible path otherwise it will be an infeasible path. As shown in figure 9[1, 10, 14, 36] is a feasible path whereas [1, 6, 36] is an infeasible path. Since the obstacle boundary consist of the actual boundary of the obstacles and the minimum safety distance for the robot, that's why the mobile robot has been considered as a point throughout this work [25]. The environment for this work has been taken as static i.e. the obstacles are remain fixed throughout the whole process.

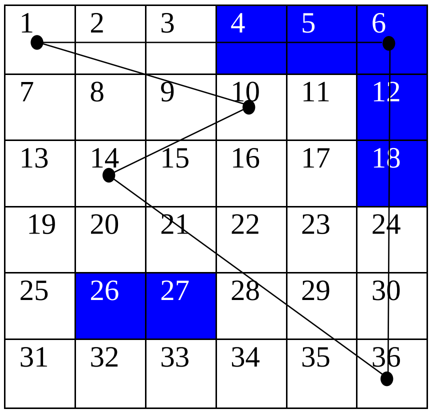


Figure 9: Robot's path in the environment

5.2 Proposed method

5.2.1 Initial population

Each of the individuals or paths in the initial population, also known as chromosomes are generated randomly in such a way that none of the individuals are infeasible. The reason is that the presence of a chromosome that represent an infeasible path should not ideally be included, more over the presence of such chromosome may reduce the convergence rate. Here each of the individuals of the initial population is generated by the way described algorithm 5.

```
1. $P_l =$ is the length of the path.
2. $P$ is the newly generated path.
3. $Free\_grid[] =$ grid number of all the blank grids in the environment.
4. $P[] =$ start node.
5. Until Free_grid not empty and length of $P < P_l - 1$:
        a. $l = P[last\ element]$
        b. $g =$ select a grid number randomly from Free_grid
        c. if $[l, g]$ do not intersect with any colored grid:
                i. $P[\,] = g$
                ii. Delete $g$ from Free_grid
6. If Free_grid is empty or length of $P < P_l - 1$:
        go to step 2
7. $l = P[last\ element]$
8. If $[l, Target\ node]$ do not intersect with any colored grid:
        $P[\,] = Target\ node$
   Else:
        go to step 2
9. Return $P$
```

Algorithm 5: Algorithm for initial population

5.2.2 Fitness

Let P be a path of size n from starting node to target node. Since distance has been considered as an optimization criteria, the objective function (f) for path P is:

$$f = \sum_{i=1}^{n} \sqrt{(x_{(i+1)} - x_i)^2 + (y_{(i+1)} - y_i)^2} + penalty \qquad 5.1$$

$$penalty = \begin{cases} Longest\ path\ segment\ possible\ according\ to\ the\ environment & if\ path\ is\ infesbible \\ 0 & if\ path\ is\ feasible \end{cases}$$

In equation-5.1 $x_{(i+1)}$and x_i are the x coordinate of the $i-th$ and $(i+1)-th$ node of path Prespectively. Similarly, $y_{(i+1)}$ and y_i are the y coordinate of the $i-th$ and $(i+1)-th$ node of path P.

Since the purpose of this path planning is to get the shortest feasible path from starting node to target node. That's why the above mentioned objective function (f) needs to be minimized.

5.2.3 Genetic Operators

There are three genetic operators. These are selection, crossover and mutation. A new crossover operator has been introduced in this work and the mutation operators is also modified a bit. These three genetic operators are described below.

5.2.3.1 Selection

Selection based on the fitness value of the chromosomes may lead to the duplication of chromosomes in the mating pool. So in this work, all the chromosomes in the population are selected in the mating pool irrespective of their fitness values.

5.2.3.2 Crossover

The crossover has been used in this work is based on the crossover used in [26]. It consists of three cases.

Case 1:

The common nodes are the nodes those are present in both parent chromosome. But sometime these common nodes are present in both of parent paths, sometimes in either one or none of them, though they are the part of both parent paths. Here in this work, crossover has been performed on all such common nodes one by one except the starting node and target node. If there are n numbers of common nodes, then n pairs of children will be generated. Out of n

such pairs, the pair having lowest average fitness value has been taken as children. For example, consider two parent chromosome or path $P1[1,2,14,22,36]$ and $P2[1,2,7,10,34,36]$ as in Figure 10.

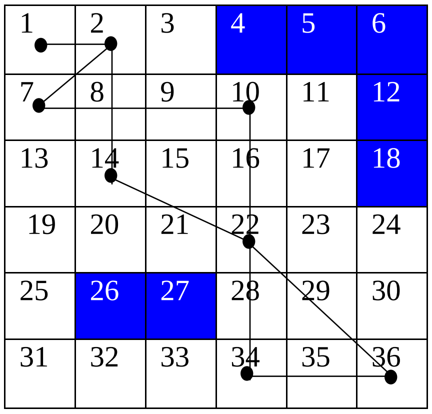


Figure 10: Example for crossover case 1

This two paths have 3 common nodes, out of them one is visible i.e. 2 since it present in both path. Others are invisible. They are 8 (present neither on $P1$ nor on $P2$) and 22 (present only on $P1$). Crossover has been done on each of the three common nodes. For this purpose, the invisible common nodes have to be inserted in the parent paths. So, after insertion $P1$ becomes $[1,2,8,14,22,36]$ and $P2$ becomes $[1,2,7,8,10,22,34,36]$.

Pair 1: (Crossover at 2)

Child 1: [1, 2, 7, 8, 10, 22, 34, 36]
Child 2: [1, 2, 8, 14, 22, 36]

Pair 2: (Crossover at 8)

Child 1: [1, 2, 8, 10, 22, 34, 36]
Child 2: [1, 2, 7, 8, 14, 22, 36]

Pair 3: (Crossover at 22)

Child 1: [1, 2, 8, 14, 22, 34, 36]
Child 2: [1, 2, 7, 8, 10, 22, 36]

Out all three pairs, the pair having the lowest average fitness value will be selected as children for parent P1 and P2.

Case 2:

An interconnected node pair refers to different node from each of the parent path or chromosome which can be connected without intersecting any colored grid. These interconnected nodes are taken as a common point. Crossover has been performed on all such nodes one by one. If there are n numbers of interconnected node pairs, then n pairs of children will be generated. Out of n such pairs, the pair having least average fitness value has been taken as children. For example, consider parent chromosome or path $P1$ [1, 9, 16, 36] and $P2$ [1, 14, 36] shown in Figure 11.

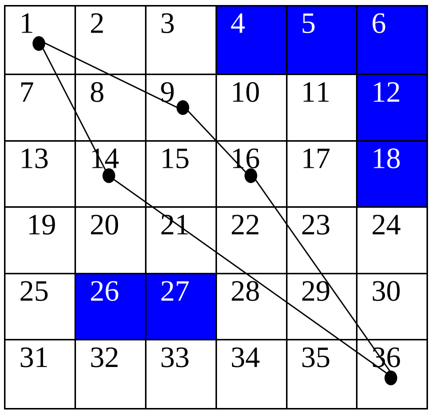


Figure 11: Example of crossover case 2

Now this parent has two interconnected node pairs [9, 14] and [16, 14] and so there will be two crossover, one for each pairs.

Pair 1: (Crossover at [9, 14])

Child 1: [1, 9, 36]
Child 2: [1, 14, 16, 36]

Pair 2: (Crossover at [16, 14])

Child 1: [1, 9, 16, 36]
Child 2: [1, 14, 36]

Out of pair 1 and pair 2, the pair having lowest average fitness will be the children.

Case 3:

Parent chromosomes are interchanged with each other. For example, consider parent $P1$ [1, 31, 36] and $P2$[1, 21, 36] as shown as Figure 12.

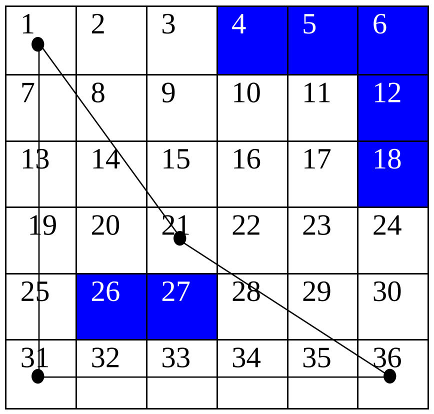


Figure 12: Example for crossover case 3

So, after crossover:
Child 1: [1, 21, 36]
Child 2: [1, 31, 36]
Algorithm 6 is the algorithm for crossover.

```
1. $P1$ and $P2$ are two parent chromosome.
2. Let $C1$ and $C2$ are the child.
3. Find out all the common nodes (visible as well as invisible) between $P1$ and $P2$ except
starting and ending node.
4. If any common node exists between $P1$ and $P2$ (except start node and target node):
        a. Perform crossover at each common nodes.
        b. Find out the average fitness value for each pair created after the above step a.
        c. Take pair having minimum average fitness value and assigned them to $C1$ and
        $C2$.
   Else:
          a. Find out all the interconnected node pair between $P1$ and $P2$
          b. If any interconnected node pair exists between $P1$ and $P2$:
                i. Perform crossover at each interconnected node pairs.
                ii. Find out the average fitness value for each pair created after the above
                step $i$.
                iii. Take the pair having minimum average fitness value and assigned them
                to $C1$ and $C2$.

             else:
                i. interchange $P1$ and $P2$ and assigned to $C1$ and $C2$.
5. Return $C1$ and $C2$.
```

Algorithm 6: Algorithm for Crossover

5.2.3.3 Mutation

```
1. $P_{mut}$ is Predefined mutation probability.
2. $P$ is Path or chromosome.
3. For each node $P_i$ in $P$ (except starting node and target node):
        a. $P_{rand} =$ random number in the interval $[0, 1]$.
        b. if $P_{rand} \leq P_{mut}$ :
                i. Replace the node $P_i$ with its neighbor grids (except the colored grid and
                the nodes belongs to path $P$) and find out the path distance for each path
                (including the previous path having node$P_i$ ).
                ii. Replace $P_i$ with the grid causing minimum fitness.
4. Return $P$
```

Algorithm 7: Algorithm for mutation

The only difference between mutation operator used in this work and mutation operator used in [27] is that, in [27] only one node (except start and goal node) of the path get a chance to perform mutation with some predefined mutation probability whereas in this work each node of the path (except start and goal node) get a chance to perform in mutation with some predefined mutation probability. The algorithm for mutation has been stated in algorithm 7.

For example, consider a path P [1, 21, 36] in Figure 13 where node 21 is selected for mutation. For this purpose, the neighbors of 21 are 20, 14, 15, 16, 22, 28, 27, and 26. So, 21 may be replaced by one of the neighbors except 27 (since it is a colored grid or obstacle) that produce least fitness value among them (including 21 also).

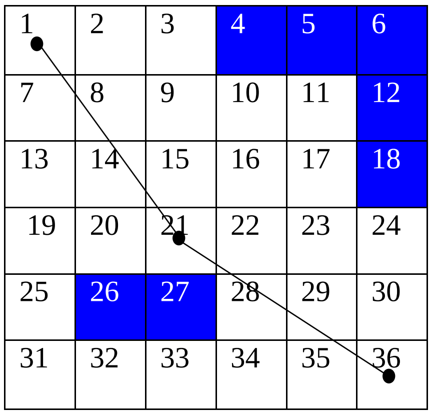


Figure 13: Example for mutation

5.2.4 Domain Knowledge Based Operators

Some Domain Knowledge Based Operators have also been used in this work along with the genetic operators. These are Circuit Removal operator, Insertion-Deletion operator and Refinement operator. As the name suggests these operators are based on the domain knowledge. Not only that they also incorporates small-scale local search that improves the efficiency of the operators [25]. They are described below.

5.2.4.1 Circuit Removal

Circuit Removal has been used in this work is different from regular circuit removal techniques. Usually, if there is multiple occurrences of nodes in a chromosome, then circuit occurs. But here in this scenario, sometimes though there is no repetition of nodes in a chromosome, it may contain some circuits. So, depending upon this, circuits are of three types. They are visible circuit, hidden circuit and combination of visible and hidden circuit.

Visible Circuit

If circuit occurs only due to the repetition of nodes in the chromosome, then it will be considered as a visible circuit. For example, consider the chromosome [1, 3, 9, 15, 8 ,9, 23, 36] as in Figure 14. Here circuit occurs only due to the multiple occurrence of node 9.

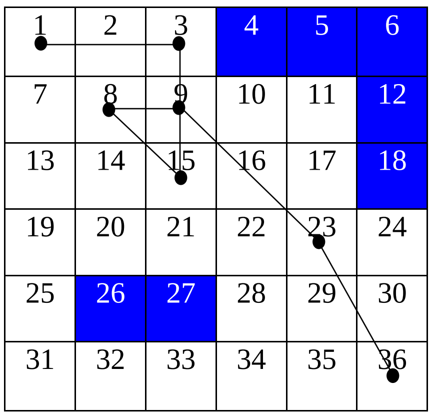


Figure 14: Example of visible circuit

Hidden Circuit

Sometimes though there is no repetition of any node in the chromosome, but it may contain some circuit. This type of circuits are again two types.

Type 1:

Circuit due to the intersection between two nonadjacent path segments.

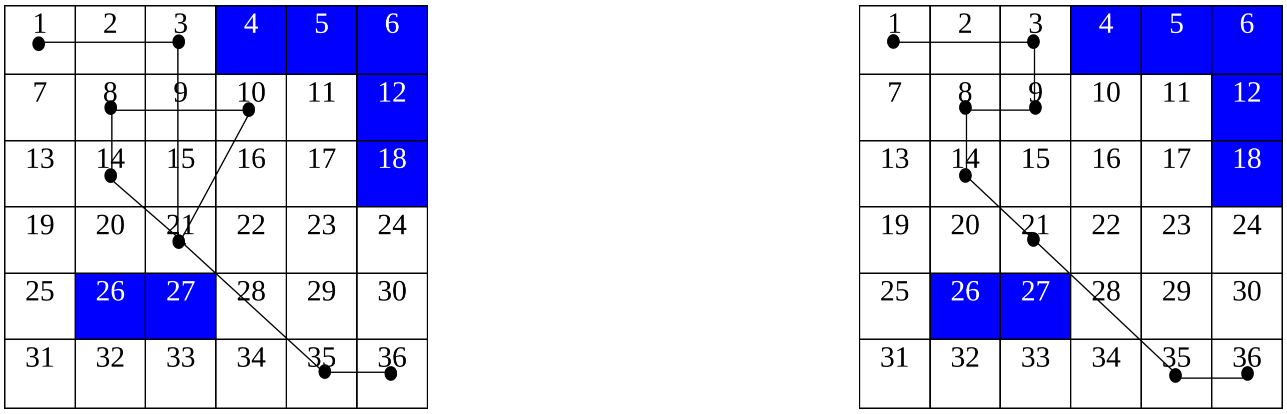
Type 2:

A path segment passes through a node that is also belongs to that path segment.

For example: consider the path $[1\,,3,21,10,8,14,35,36]$ in figure 15 (a). Though there is no repetition of any nodes, it contains circuits two circuits of each of types described above. The circuit of type 1 is due to intersection between two nonadjacent path segments $[3,21]$ and $[10,8]$ and another circuit of type 2 is created since path segment $[14,35]$ passes through a node 21 which is also belongs to the path segment $[14,35]$. After applying Circuit Removal the path become $[1,3,9,8,14,21,35,36]$ as shown in figure 15 (b).

Combination of Visible and Hidden circuit

This type of circuit occurs due the multiple occurences of nodes as well as due to the hidden circuits.



(a) (b)

Fig. 15. (a) Path before Circuit removal and (b) Path after applying Circiut Removal.

5.2.4.2 Insertion-Deletion Operator

Deletion operator select two adjacent path segment and check that the starting node of the first segment and the last node of the second segment can be joined without having obstacle in between them. For instance, let $[P_i,\ P_j]$ and $[P_j,\ P_k]$ are two adjacent path segment of path P and then delete node P_j, if P_i and $P_k$can be connected with out colliding with any obstacle or colored grid. Utility of this deletion operator is that it increases the convergence rate. But sometimes, application of Deletion operator results in a path having very few number of nodes that causes premature convergence. That's why insertion operator has been introduced here. Insertion operator insert a node in any of the path segment randomly. After insertion the inserted node is fitted into proper position after applying other operators on it. Algorithm 8 describes the algorithm for insertion-deletion.

```
1. $P$ is the path
2. Select insertion or deletion with $.50$ probability.
3. If insertion is selected:
        a. $P_I$ is the probability for insertion.
        b. For all line segment in $P$ :
                i. $r =$ generate a random number in the range $[0, 1]$.
                ii. If $r \leq P_I$ :
                    • Randomly select a node in between the selected path segment.
                    • Insert the node in between the selected line segment in the path $P$.

4.  If deletion is selected:
        a. $P_d =$ probability for deletion.
        b. For all two adjacent line segment in $P$ :
                // Let  $[P_i,\ P_j]$and $[P_j,\ P_k]$ are two adjacent path segment.

                i. $r =$ generate a random number in the range $[0, 1]$.
                ii. If $r \leq P_d$ :
                    • If $[P_i,\ P_k]$is a path segment which does not intersect with any
                      colored grid:
                          Delete $P_j$ from the Path $P$.
5. Return path P.
```

Algorithm8: Algorithm for Insertion-Deletion

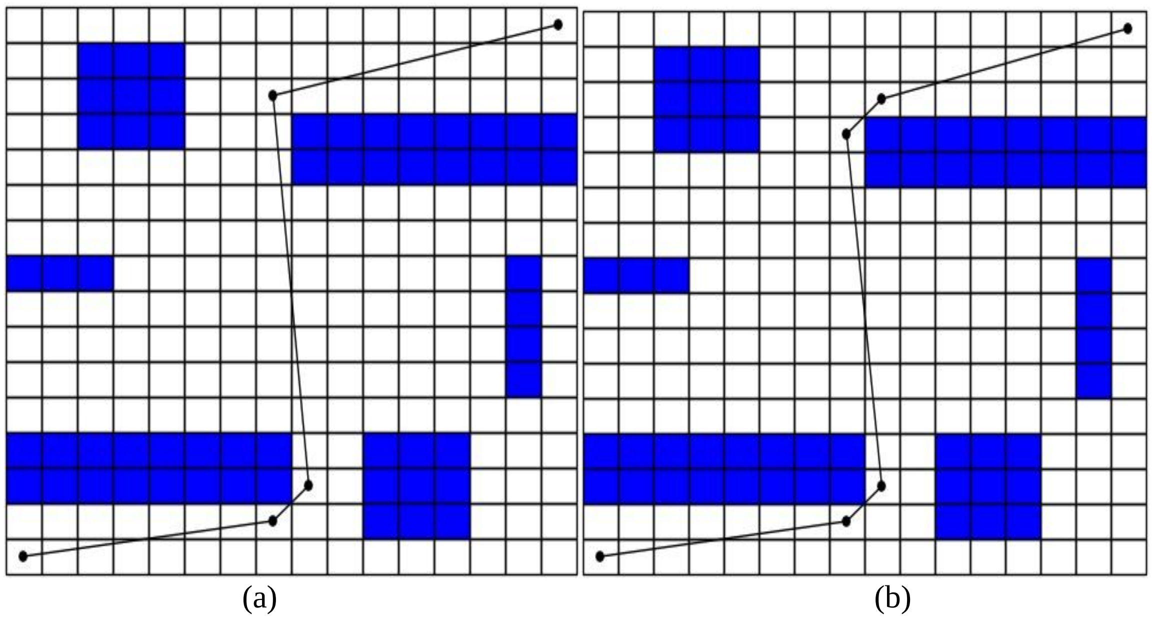


Figure 16. (a) Path before Insertion and (b) Path after applying Insertion.

For example consider, the environment shown in Figure 16 (a) and (b). The path shown in Figure 16 (a) is a near optimal path and after applying inserting operator which introduces an additional node in the path as a result the path is converted into optimal path as shown in Figure 16 (b).

5.2.4.3 Refinement operator

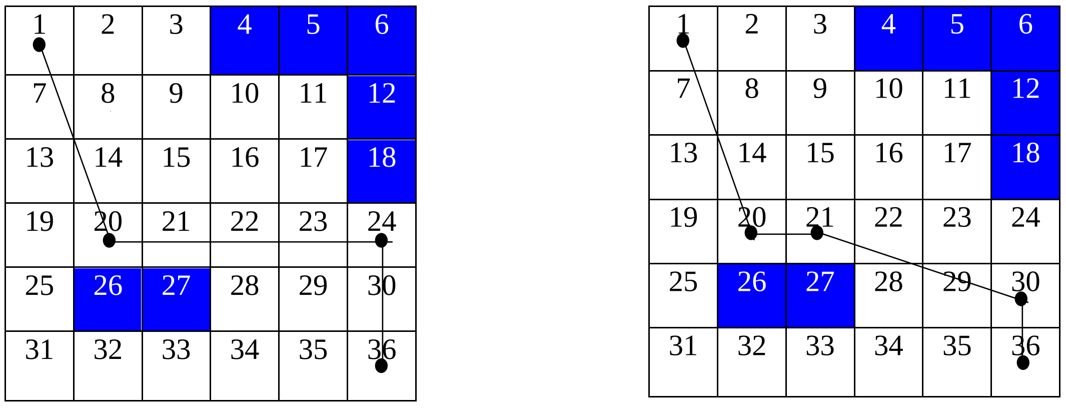
The refinement operator has been used in this work is based on the refinement operator used in [26]. It is based on the principle that the length of hypotenuse in a right angle triangle must be less than the summation of the lengths of other two sides. Algorithm 9 describes the algorithm for this Refinement Operator.

```
1. Let $P$ be a path.
2. For all two adjacent line segment in $P$ :
        // Let $[P_i,\ P_j]$ and $[P_j,\ P_k]$ are two adjacent path segment.

        a. If these two path segment create a right angle:
            i. If $[P_i,\ P_k]$do not intersect with any colored grid:
                ▪ Delete node $P_j$ from the path.
            ii. Else:
                ▪ Find all the intermediate node between the path segment $[P_i,\ P_j]$
                  and insert them in the proper order starting from $P_{i+1}$ to$P_j$
                  i.e.$[P_{i+1}, \dots, P_j]$.
                ▪ Find all the intermediate node between the path segment $[P_j,\ P_k]$
                  and insert them in the proper order starting from $P_{k+1}$to $P_j$ i.e.
                  $[P_{k+1}, \dots, P_j]$.
                ▪ $X_i = P_{i+1}$.
                ▪ $X_j = P_{j+1}$.
                ▪ Until $X_i \neq P_j$ and $X_j \neq P_j$ :
                        ➢ If $[X_i,\ X_j]$ do not intersect with any colored grid:
                            ❖ Delete $P_j$ from path $P$.
                            ❖ Insert $X_i$ after $P_i$ in path $P$.
                            ❖ Insert $X_j$ before $P_k$ in path $P$.
                        ➢ Else:
                            ❖ $X_i = X_{i+1}$
                            ❖ $X_j = X_{j+1}$
3. Return Path P
```

Algorithm 9: Algorithm for refinement Operator

For example, consider path $[1, 20, 24, 36]$ as in Figure 17 (a). In this path two adjacent path segments $[20, 24]$ and $[24, 36]$ create a right angle. Since path segment $[20, 36]$ collides with obstacles. So, after inserting all the intermediate node in this two path segment become $[21, 22, 23, 24]$ and $[30, 24]$. Now, since $[21, 30]$ don't collide with any obstacles, at first delete 24 from the path. Then insert 21 after 20 and 30 before36. As a result, the path become $[1, 21, 30, 36]$ shown in Figure 17 (b).



(a) (b)

Figure 17. (a) Path before refinement operator and (b) Path after applying refinement operator.

5.2.5 Elitist Strategy

The purpose of this method is to keep the best chromosome of the previous generationg_i into the next generationg_{i+1}. The algorithm is described in Figure 17which is based on the elitist strategy in [53].

```
1. Let $B_i$ be the best chromosome in the generation $g_i$.
2. $e$ be the elite chromosome.
3. Let $W_i$ be the worst chromosome in the generation $g_i$.
4. $e = B_0$ (the best chromosome of the initial population i.e $g_0$).
5. $n$ be the total no of iterations or generations.
6. For $i = 1$ to $n$:

    a. Perform Selection, Crossover, Circuit Removal, Mutation, Insertion-Deletion
       Operator, Path Refinement Operator on the population of generation $g_{i-1}$ to
       obtain new population of generation $g_i$.
    b. If $W_i$ is worse than $B_{i-1}$ in terms of their fitness value:

        i. Replace $W_i$ by $B_{i-1}$.

    c. $e = B_i$
```

Algorithm 10: Algorithm for Elitist Strategy

5.2.7 Termination Condition

Though there is no such stopping criteria for the GA [6], the algorithm will be terminated when either one of the following three conditions will be satisfied:

1. The total no of generation exceeds 100.
2. The algorithm converges to the optimal solution with in 100 numbers of generation.
3. The fitness value of the best chromosome remain unchanged for around 50 consecutive generations.

5.3 Flowchart

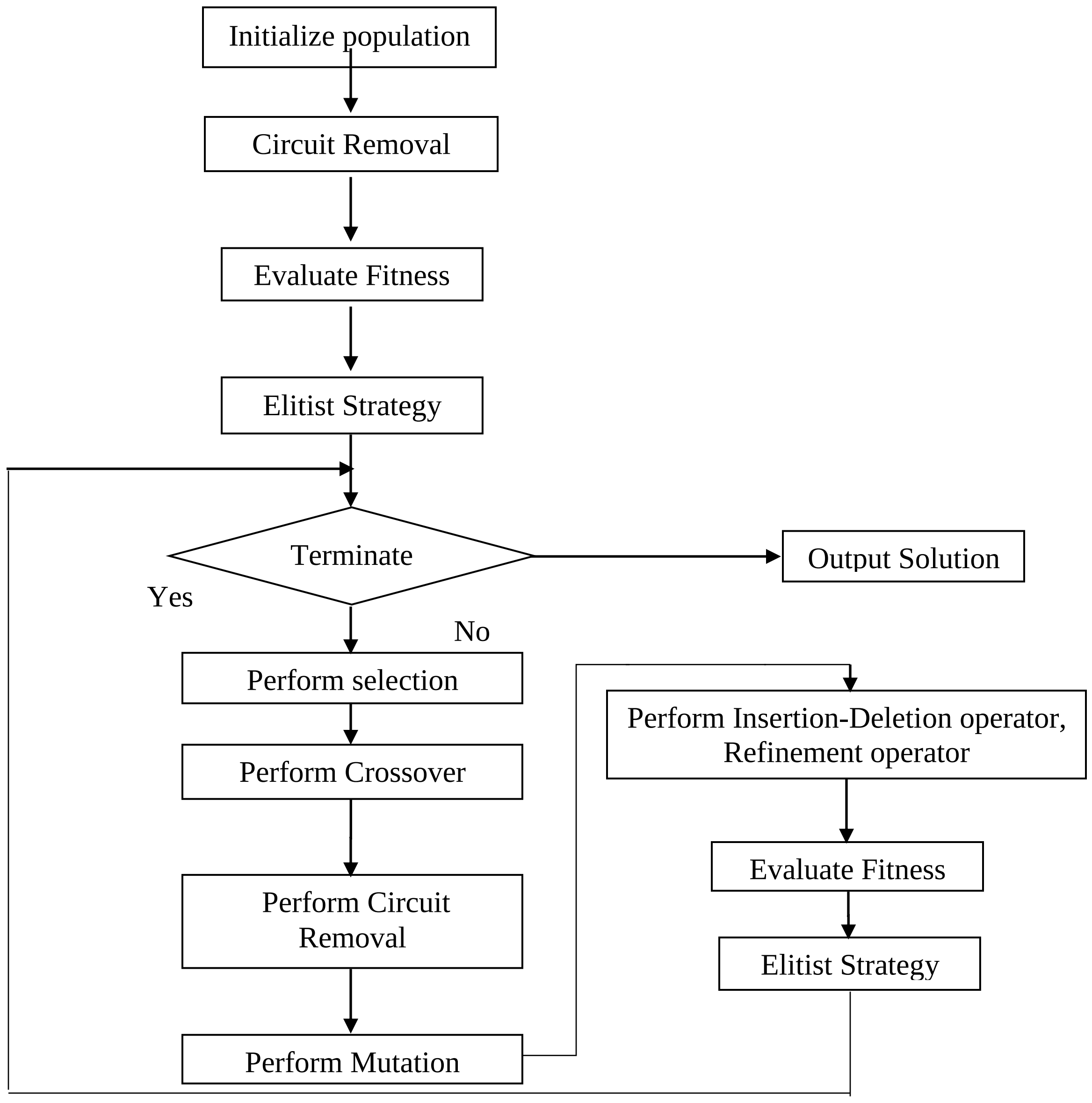


Figure 18: Flow chart of the proposed algorithm.

The flowchart of the entire method has been shown in the above figure 18.

5.4 Experimental Results

Simulation has been done on a computer having i5 2nd generation processor and 4 GB RAM using Python 2.7 language. We have simulated several environments for implementation of our proposed method. It may be noted that these environments are also used on some previous work [27, 26, 28] on this topic too. This has facilitated the comparison of our experimental result with that of the said previous work to solve the same problem. It has been also applied on some new environments that are not considered in any previous works. The values of different operators used in all the experiments are as follows: Mutation probability-0.25, Crossover probability-1.0, Circuit Removal probability-1.0, Insertion-Deletion probability- 0.30 for Deletion and 0.20 for Insertion and Refinement operator probability-1.0. All the results in the following tables are the average of 100 experimental runs.

Figure 19(a) represents environment 1, population size for this environment is 10. Table 1 shows the results.

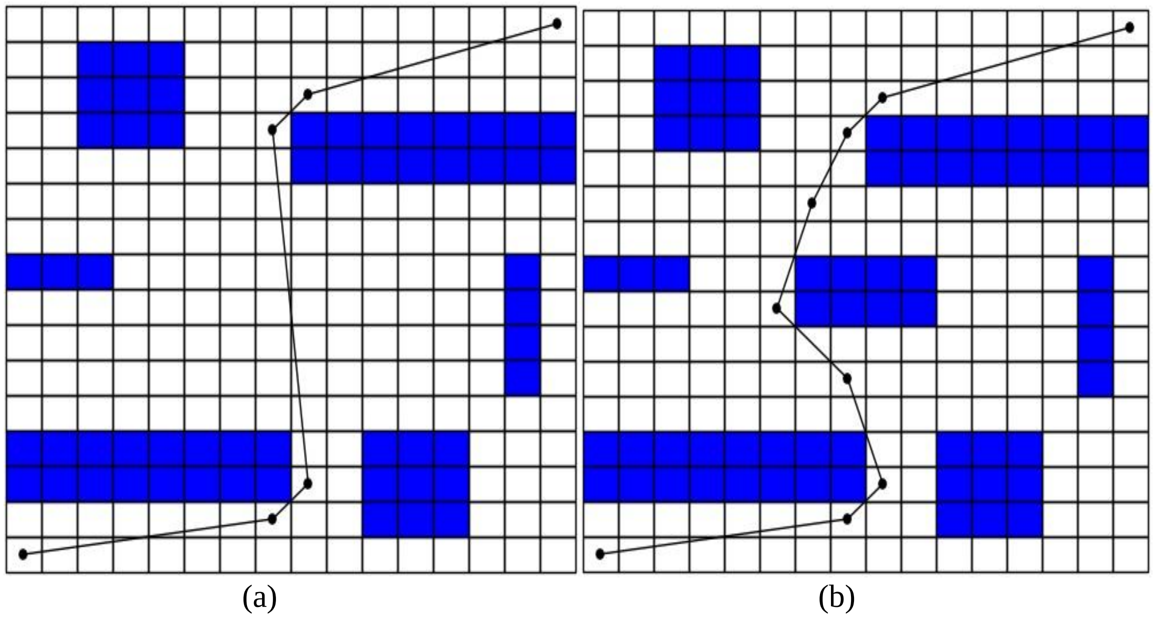


Figure 19: (a) environment 1 and (b) environment 2.

Table 1: (For the environment shown in Fig 19(a))

	# of optimal solution	# of near optimal solution	# of infeasible solution	Fitness value	# of generations	Time to converge (s)
Reference [26]	3	69	28	29.25	23	0.31
Reference [27]	54	44	2	27.82	11	0.89
Reference [28]	*[1]	*	0	27.68	136	4.07
Proposed Approach	100	0	0	27.22	18	3.13

It may be noted from table 1 that the proposed method is able to find the optimal solution in all the hundred runs on the environments shown in Figure 19(a). The average fitness value, the number of iteration / generation of the method and the time to converge needed to obtain the optimal solution are 27.22, 18 and 3.13 sec respectively. It may be noted that the average fitness value obtained by using the proposed method is less than that of the other three methods and the time to converge is not also very high.

Table 2 represents the results for environment 2 as shown in Figure 19(b). The population size for this environment is taken as 30.

Table 2: (For the environment shown in Fig 19(b))

	# of optimal solution	# of near optimal solution	# of infeasible solution	Fitness value	# of generations	Time to converge (s)
Reference [26]	0	95	5	31.21	22	0.26
Reference [27]	44	56	0	29.08	11	0.86
Reference [28]	*[2]	*	0	28.87	132	3.62
Proposed Approach	96	4	0	28.56	41	28.57

[1] These results are not reported in [28].
[2] These results are not reported in [28].

In this case also, our proposed method has obtained the lowest average value for fitness function and also provided optimal solutions ninety six times out of hundred times. It is obvious that the experimental result obtained by our proposed method is much better than that of the rest three methods. Only the time to converge is on the higher side.

The environment 3 is represented in Figure 20(a). The result for this environment is given in Table 3. The population size for this environment is taken to be 20.

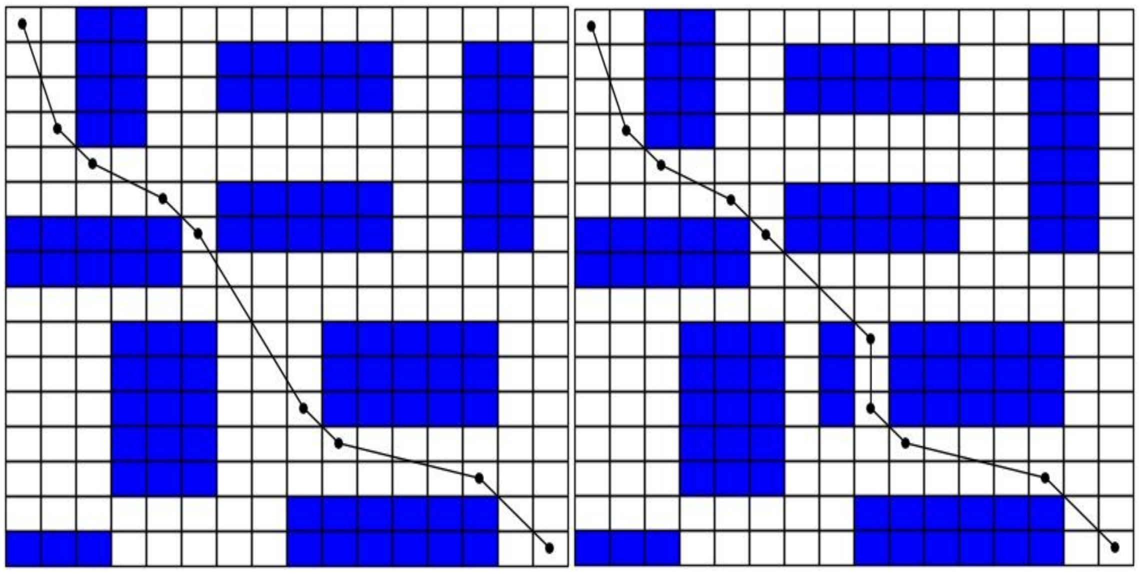


Figure 20: (a) environment 3 (b) environment 4.

Table 3: (For the environment-3 shown in Fig 20 (a))

	# of optimal solution	# of near optimal solution	# of infeasible solution	Fitness value	# of generations	Time to converge (s)
Reference [26]	1	68	31	25.02	65	1.03
Reference [27]	9	78	13	24.68	16	1.68
Reference [28]	*[3]	*	0	23.40	111	4.07
Proposed Approach	100	0	0	22.42	17	7.58

Here the proposed method has found the optimal solutions all the hundred times. The average fitness value, the number of iteration / generation of the method and the time to

[3] These results are not reported in [28].

converge needed to obtain the optimal solution are 22.42, 17 and 7.58 sec respectively. Note that the average fitness value is again the lowest one compared to that of other methods.

Figure 20 (b) presents the environment-4 and the result obtained for this environment is given in Table-4. The population size for this environment is taken to be 40.

In this experiment also, the proposed method is successful to find the lowest average value of the fitness function. It may be noted that the proposed method has found the optimal solution fifty three times out of hundred times, whereas the same figure for the other two methods are six and thirty two respectively.

Table 4: (For the environment-4 shown in Fig 20 (b))

	# of optimal solution	# of near optimal solution	# of infeasible solution	Fitness value	# of generations	Time to converge (s)
Reference [26]	6	92	2	25.17	31	0.34
Reference [27]	32	68	0	24.71	12	0.69
Reference [28]	*[4]	*	0	23.59	101	3.62
Proposed Approach	53	47	0	23.21	18	16.54

It may be noted that for all the experiments mentioned above, the proposed method is able to reach the optimal solution either all the time or most of the time out of hundred runs and it is able to provide the lowest average value of fitness function.

The proposed method has also been applied to one new environment which is not considered by any previous works on this topic which is Environment-5 is shown in Figure 21 and the population size for this environment is taken to be 10. The results for environments 5 is presented in Table 5.

[4] These results are not reported in [28].

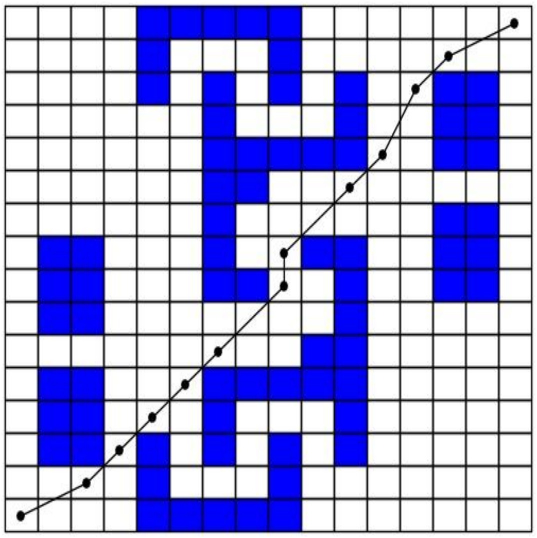


Figure 21: environment 5

Table 5 (for the environments shown in Fig. 21)

	# of optimal solution	# of near optimal solution	# of infeasible solution	Fitness value	# of generations	Time to converge (s)
Environment 5	95	5	0	21.95	11	9.48

Note that here the proposed method is successful in ninety five times out of hundred times runs to optimal solution. So, it seems that the proposed method is efficient enough to reach optimal solution most of the times for any arbitrary environments also.

Chapter 6: Conclusion and scope for future work

6.1 Conclusion

In this work an improved genetic algorithms based techniques has been developed to deal with robot path planning problem where the objective is to find a collision free optimal path. In this work, the length of the chromosome has been taken as variable where the length varies from two i.e. for only the start node and target node to $(total\,number\,of\,grids - total\,number\,of\,obstacles)$. Since, it is very difficult for an environment to anticipate the appropriate size of chromosome. Also in case of population size, it is noteworthy that there exist no such guideline for choosing the appropriate value of the population size [53]. Nevertheless, as the population size increases, the searching process become more diverse, but it introduces some extra computational effort. In this work, different population size has been considered for different environment where the population size has been chosen experimentally i.e. by taking appropriate one after applying different population size on each of the environments. It seems that there is a relationship between the population size taken and the shape, size and location of the obstacles present in a given environment. In this work, some domain knowledge based operators have also been used along with the regular GAs which perform some local search in order to increase the efficiency of the searching process. Based on the experimental results, it can be concluded that the method used in work performs better than some previously developed approaches [26 - 28] for finding collision free optimal path in robot path planning problem and has provided better result in terms of finding both optimal solution and the average value of the fitness function. Though for some environment the execution time is a bit high. Above all the proposed approach has succeed to converge into optimal solution hundred out of hundred times for some environment which none of the aforementioned approaches can able to find.

6.2 Scope for future work

Here in this work the environment has been taken as static i.e. the obstacles will not change throughout the whole process. So, path planning in dynamic environment can be considered as one of the possible future work where the obstacles may change their position where the obstacles are not fixed. For instance the obstacles may change their positions and / or some new obstacles may appear / disappear as time goes on. Our concept can be extended for path planning of multiple mobile robots. Parallel genetic algorithm can be used for this purpose where there will be different population for every robot and after applying all the operators on each of the population, it is required to check whether the robots are colliding with themselves or not.

Appendix 1

Research Article Communications on the basis of the work done for this Thesis:

Ritam Sarkar and Nirmalya Chowdury, An Improved Genetic Algorithm Based Path Planning for Mobile Robot (Communicated).

Appendix 2

References

[1]https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/history.html

[2] https://en.wikipedia.org/wiki/Robot

[3] https://en.wikipedia.org/wiki/Outline_of_robotics

[4] Stuart K. Card, Thomas P. Moran and Allen Newell, The Keystroke-level model for user performance time with interactive systems, Magazine Communications of the ACM 23 (1980) 396 – 410

[5] P. Raja and S. Pugazhenthi, Optimal path planning of mobile robots: A review, International Journal of Physical Sciences 7 (2012) 1314 – 1320.

[6] Roland Siegwart and IIIah R. Nourbakhsh, Introduction to Autonomous Mobile Robot, The MIT Press, Cambridge (2004).

[7] Nikolaus Correll, Introduction to Autonomous Robots, Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported (2016).

[8] Enric Galceran and Marc Carreras, A survey on coverage path planning for robotics, Robotics and Autonomous Systems 61 (2013) 1258 – 1276.

[9] Vladimir J. Lumelsky, Snehasis Mukhopadhyay and Kang Sun, Dynamic Path Planning in Sensor –Based Terrain Acquisition 6 (1990) 462 – 472.

[10] H. Choset, P. Pignon, Coverage path planning: the boustrophedon cellular decomposition, in: Proceedings of International Conference on Field and Service Robotics (1997).

[11] E.U. Acar, H. Choset, A.A. Rizzi, P.N. Atkar, D. Hull, Morse decompositions for coverage tasks, International Journal of Robotics Research 21 (4) (2002) 331–344.

[12] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers (1991).

[13] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementation, The MIT Press (2005).

[14] H. Choset, E. Acar, A.A. Rizzi, J. Luntz, Exact cellular decompositions in terms of critical points of Morse functions, in: Proc. IEEE Int. Conf. Robotics and Automation ICRA'00 3 (2000) 2270–2277.

[15] E. Acar, H. Choset, Sensor-based coverage of unknown environments: incremental construction of morse decompositions, International Journal of Robotics Research 21 (4) (2002) 345–366.

[16] H. Choset, Coverage of known spaces: the boustrophedon cellular decomposition, Autonomous Robots 9 (3) (2000) 247–253.

[17] L. Xu, Graph Planning for Environmental Coverage, Ph.D. Thesis, Carnegie Mellon University (2011.

[18] Piero P. Bonissone, Soft computing: the convergence of emerging reasoning technologies 1 (1997) 6 – 18.

[19] Lofti A. Zadeh, Soft Computing and Fuzzy Logic, IEEE Software 6 (1994) 48 – 56.

[20] Lofti A. Zadeh, Fuzzy Sets, Information and Control 8 (1965) 338 - 353.

[21] https://en.wikipedia.org/wiki/Fuzzy_logic

[22] F. Rossenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychological Review 65 (1958) 386 – 408.

[23] Bernard Widrow and Marcian E. Hoff, Adaptive switching circuits (1960).

[24] Ellips Masehian and Davoud Sedighizadeh, Classic and Heuristic Approaches in Robot Motion Planning – A Chronological Review, International Scholarly and Scientific Research & Innovation 1 (5) (2007) 228 - 233.

[25] Yanrong Hu and Simon X. Yang, A knowledge Based Genetic Algorithm for Path Planning of Mobile Robot, in: Proc.of Int. Conf. on Robotics & Automation (2004) 4350 – 4354.

[26] Qing Li, Wei Zhang, Yixin Yin, Zhiliang Wang and Guangjun Liu, An Improved Genetic Algorithm of Optimum Path Planning for Mobile Robots, in: Proc. Of Sixth Int. Conf. on Intelligent Systems Design and Applications (2006)

[27] Adem Tuncer and Mehmet Yildirim, Dynamic path planning of mobile robots with improved genetic algorithm, Computer and Electrical Engineering 38 (2012) 1564 – 1572.

[28] Amir Hossein Karami and Maryam Hasanzadeh, An adaptive genetic algorithm for robot motion planning in 2D complex environments, Computer and Electrical Engineering 43 (2015) 317 – 329.

[29] Ching-chih Tsai and Hsu-Chih Huang, Parallel Elite Genetic Algorithm and Its Application to Global Path Planning for Autonomous Robot Navigation, IEEE Trans. On Industrial Electronics 58 (10) (2011) 4813 – 4821.

[30] Zhang Qiaorong, Gu Guochang and Zhang Qiaorong, Path Planning Based on Improved Binary Particle Swarm Optimization Algorithm, IEEE Conference on Robotics, Automation and Mechatronics (2008) 462 – 466.

[31] Tan Guan-Zheng, He Huan and Sloman Aaron, Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots, Acta Automatica Sinica 33 (3) (2007) 279 – 285.

[32] P. Raja and S. Pugazhenthi, Path Planning for Mobile Robots in Dynamic Environment using Particle Swarm Optimization, Int. Conf. on Advances in Recent Technologies in Communication and Computing (2009) 401 – 405.

[33] M. A. Porta Garcia, Oscar Montiel, Oscar Castillo, Roberto Sepulveda and Patricia Melin, Path Planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost evaluation function evaluation, Applied Soft Computing 9 (2009) 1102 – 1110.

[34] Chengyu Hu, Xiangning Wu, Qingzhong Liang and Yongji Wang, Autonomous Robot Path Planning Based on Swarm Intelligence and Stream Functions, Springer-Verlag Berlin Heidelberg (2007) 277 – 284.

[35] Christopher Kozakiewicz and Masakazu Ejiri, Neural Network approach to Path Planning for Two Dimensional Robot Motion, IEEE/RSJ Int. Workshop on Intelligent Robota and Systems, Osaka, Japan (1991)

[36] Fan Jian, Fei MinRui and Ma ShiWei, RL-ART2 Neural Network Based Mobile Robot Path Planning, Porc. Of the Sixth Int. Conf. to Intelligent Systems Design and Applications (2006).

[37] R. Glasius, A. Komoda and S. Gielen, Neural network dynamics for path planning and obstacle avoidance, Neural Networks (1994).

[38] Hong Qu, Simon X, Allan R. Willms and Zhang Yi, Real-time Robot Path Planning Based on a Modified Pulse Coupled Neural Network Model, IEEE Trans. On Neural Networks 20 (11) (2009) 1724-1739.

[39] Fedor A. Kolushev and Alexender A. Bogdanov, Multi-agent Optimal Path Planning for Mobile Robots in Environment with Obstacles, Springer-Verlog Berlin Heidelberg (2000) 503 – 510.

[40] Peter Svestka and Mark H. Overmars, Coordinated path planning for multiple robots, Robotics and Autonomous System 23 (1998) 125 – 152.

[41] Wang Mei and Wu Tie-jun, Cooperative co-evolution based distributed path planning of multiple mobile robots, journal of Zhejiang University SCIENCE (2005) 697-706.

[42] Hong Qu, Ke Xing and Takacs Alexander, An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots, Neurocomputing 120 (2013) 509 – 517.

[43] Guanghui Li, Atsusji Yamashita, Hajime Asama and Yusuke Tamura, An Efficient Improved Artificial Potential Field Based Regression Search Method for Robot Path Planning, IEEE Int. Conf. on Mechatronics and Automation (2012) 1227 – 1232.

[44] S. S. Ge and Y. J. Cui, Dynamic Motion Planning for Mobile Robots Using Potential Field Method, Autonomous Robots 13 (2002) 207 – 222.

[45] Hossein Adeli, M. H. N. Tarbrizi, Alborz Mazloomian, Ehsan Hajipour and Mehran Jahed, Path Planning for Mobile Robots using Iterative Artificial Potential Field Method, IJCSI Int. Journal of Computer Science 8 (4) (2012) 28 – 32.

[46] Allan R. Willms and Simon X. Yang, An Efficient Dynamic System for Real-Time Robot-Path Planning, IEEE Trans. On Systems, Man and Cybernetics, Part B: Cybernetics 36 (4) (2006) 755 – 765.

[47] Meng Wang and James N. K. Liu, Fuzzy Logic Based Robot Path Planning in Unknown Environment, Proc. Of the Fourth Int. Conf. on Machine Learning and Cybernetics, Guangzhou (2005) 813 – 818.

[48] Hui Miao and Yu-Chu Tian, Robot Path Planning in Dynamic Environments Using a Simulated Annealing Based Approach, 10th Int. Conf. on Control, Automation, Robotics and Vision Hanoi, Vietnam (2008) 1253 – 1258.

[49] H. Choset, P. Pignon, Coverage path planning: the boustrophedon cellular decomposition, in: Proceedings of International Conference on Field and Service Robotics (1997).

[50] Kuo-Chin Fan and Po-Chang Lui, Solving Find Path Problem in Mapped Environments Using Modified A^* Algorithm, IEEE Trans. On Systems, Man And Cybernetics 24 (9) (1994) 1390 – 1396.

[51] C. A. Ankerbrandt, B.P. Unckles and EE. Petry, Scene recognition using genetic algorithms with semantic nets, PatternRecognition Lett.11 (1990) 285 – 293.

[52] S. Bomholdt and D. Graudenz, Genetic asymptotic and neural networks and structure design by genetic algorithms, Neural Networks5 (1992) 327 – 334.

[53] C. A. Murthy, Nirmalya Chowdhury, In search of optimal clusters using genetic algorithms, Pattern Recognition Letters 17 (1996) 826 – 832.

[54] https://en.wikipedia.org/wiki/John_Henry_Holland

[55] S. N. Sivanandam and S. N. Deepa, Introduction to Genetic Algorithms, Springer-Verlag Berlin Heidelberg (2008).

[56] D. Bhandari, C.A. Murthy and S.K. Pal, Genetic algorithm with elitist model and its convergence, lnternat. J. PatternRecognition Artificial Intelligence (1996).

[57] Noraini Mohd Razali and John Geraghty, Genetic Algorithm Performance with Different Selection Strategies in Solving TSP, Proceedings of the World Congress on Engineering (2011).

[58] R. A. Caruana, L. A. Eshelmann and J. D.Schaffer, Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover, Eleventh International Joint Conference on Artificial Intelligence, Sridharan, N. S. (Ed.), San Mateo, California, USA: Morgan Kaufmann Publishers, 1 (1989) 750 – 755.

[59] L. *Booker*, Improving search in genetic algorithms, In [Dav87] (1987) 61 – 73.

[60] T. Yalcinoz and Halis Altun, A new genetic algorithm with arithmetic crossover to economic and environmental economic dispatch, International journal of engineering intelligent systems for electrical engineering and communications (2005).

[61] L. Davis, Handbook of Genetic Algorithms, New York, Van Nostrand Reinhold (1991).

[62] D.E.Goldberg. and R.Lingle, Alleles, loci and the travelling salesman problem, Proceedings of an International Conference on Genetic Algorithms, Morgan Kauffman, (1985) 10 – 19

[63] Nitasha Soni and Tapas Kumar, Study of Various Crossover Operators in Genetic Algorithms, Int. journal of Computer Science and Information Technologies 5 (6) (2014) 7235 – 7328.

[64] Kusum Deep and Hadush Mebrahtu, Variant of partially mapped crossover for the Travelling Salesman problems, Int. Journal of Combinatorial Optimization and Informatics 3 (2012) 47 – 69.

[65] Enrique Alba and Jose M. Troya, A survey of Parallel Distributed Genetic Algorithms, Complexity 4 (4) (1999) 31 – 52.

[66] Erick Cantú-Paz, A survey of parallel genetic algorithms, Calculateurs paralleles, reseaux et systems repartis 10 (2) (1998) 141-171.

[67] Erick Cantú-Paz, Migration policies, selection pressure, and parallel evolutionary algorithms, Journal of heuristics 7 (4) (2001) 311 – 334.

[68] https://en.wikipedia.org/wiki/Evolutionary_computation

[69] D. B. Fogel, what is evolutionary computing?, IEEE Spectrum 37 (2) (2000) 28 – 32.

[70] Mitchell, Melanie, and Charles E. Taylor, Evolutionary computation: an overview, Annual Review of Ecology and Systematics 30 (1999) 593 – 616.

[71] Ajith Abraham, Handbook of Measuring System Design, edited by Peter H. Sydenham and Richard Thorn, John wiley & Sons (2005).

[72] http://www.obitko.com/tutorials/genetic-algorithms/encoding.php

[73] Eva Volna, Introduction to Soft Computing, 1st edition (2013).