# An Efficient Algorithm for Path Planning and Task Assignment of Multi-Mobile Robot System using ANN

Thesis submitted in partial fulfillment of the requirement for the award of degree of

**Master of Engineering**

**in**

**Electronics Instrumentation and Control Engineering**


By:

**Apra Gupta**

**(80651002)**

Under the supervision of:

**Dr. Yaduvir Singh, Associate Professor**

ELECTRICAL AND INSTRUMENTATION ENGG. DEPARTMENT

THAPAR UNIVERSITY


July 2008

# CERTIFICATE

This is to certify that my work presented in this thesis entitled **"An Efficient Algorithm for Path Planning and Task Assignment of Multi-Mobile Robot System using ANN"** submitted by **Ms. Apra Gupta** in partial fulfillment of the requirement for the award of the degree of **Master of Engineering in Electronics Instrumentation and Control Engineering at Thapar University, Patiala**, is an original record under supervision and guidance of **Dr. Yaduvir Singh.** The matter embodied in this report has not been submitted anywhere for the award of any degree.

Date: -

**Apra Gupta**

Roll No 80651002

It is certified that the above statement made by the student is correct to the best of our knowledge and belief.

**Dr. Yaduvir Singh**
Associate Professor, EIED
(Supervisor)
Thapar University, Patiala

**Mr. Smarajit Ghosh**
Professor& Head, EIED
Thapar University, Patiala

**Dr. R.K Sharma**
Dean of Academic Affairs
Thapar University, Patiala

Dedicated to my parents

**Shri. S.N. Gupta**

**&**

**Smt. Neelam Gupta**

**&**

**Almighty GOD**

# *ACKNOWLEDGEMENT*

*First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life.*

*I am very thankful to the Head of the Department,* ***Dr. Smarajit Ghosh****, for his encouragement, support and for providing the facilities for the completion of this thesis.*

*This work would not have been possible without the encouragement and able guidance of my supervisor* ***Dr. Yaduvir Singh.*** *Their enthusiasm and optimism made this experience both rewarding and enjoyable. Most of the novel ideas and solutions found in this thesis are the result of our numerous stimulating discussions. Their feedback and editorial comments were also valuable for writing this thesis.*

*I am also very thankful to the entire faculty and staff members of Electrical Instrumentation Department for their direct–indirect help, cooperation, love and affection, which made my stay at TIET memorable.*

*I am deeply indebted to my parents for their inspiration and ever encouraging moral support, which enabled me to pursue my studies.*

***Dated:*** ***Apra Gupta***

***(ROLL NO. 80651002)***

# ABSTRACT

A significant interest in multiple robots or robotics vehicle has been developed for last four decades. Multiple robots have been used to perform task at multiple target. For such applications, it is necessary to use various mobile robots to perform complicated task. They can complete the task in less time and can perform reliably and hence gives a work efficient system. The main problem in using multi-mobile robots is to maintain their co-ordination and cooperation among themselves. There were various approaches to control multiple robotics system. But in all those approaches there was a drawback that they could not do task assignment and motion planning simultaneously hence, there was wastage of time.

In this work, an efficient algorithm is simulated, which is based upon Self Organizing Feature of Artificial Neural Network. This algorithm is used for path planning and task assignment of multi- mobile robot system. Hence it can, control various mobile robots at various locations. The work was prompted by the requirement of time saving algorithm. Earlier algorithms do path planning and task assignment separately i.e. first task is assigned to each robot and then robot's path is planned, hence it takes more time and robots remain idle for some time.

In our Algorithm, the two things that are task assignment and path planning are done in such a way that as soon as tasks are assigned, robots starts moving, there is very less wastage of time (robot remains idle for very less time).Robots reaches to target by following cheap and less distance method. The robot closet to the target will be selected for task assignment. This approach can also, deal with changing environment like if a target requires more than one robot or if a robot breaks down.

In this work, the effect of change of learning rate has also been considered. By changing (increasing or decreasing) the learning rate, the steps of robot motion are also changed. Hence, the total path can be planned according to the requirement.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | | |
|---|---|---|
| 1. | AGV | Automated Guided Vehicles |
| 2. | ARM | Assembly Robot Arm |
| 3. | ANN | Artificial Neural Network |
| 4. | BMV | Best Matching Unit |
| 5. | BNN | Biological Neural Network |
| 6. | DOF | Degree Of Freedom |
| 7. | MDP | Markov Decision Process |
| 8. | RA | Robot Agent |
| 9. | SCARA | Selectve Compliance Assembly Robot Arm |
| 10. | SPNN | Shortest Psth Neural Network |
| 11. | SOM | Self Organising Map |
| 12. | VO | Virtual Operator |

# List of Appendices

# Literature Survey

Jorge Angeles et.al [1], studied that Trajectory planning of robot motions for continuous-path operations is formulated in configuration space resorting to the intrinsic properties of the path traced by one point of the end effector. It is shown that, by referring the orientation of the end effector to a unique orthogonal frame defined at every point of the aforementioned path, a systematic procedure for trajectory planning in configuration space was derived. The computations required determining the angular velocity and angular acceleration of the path frame reduced to computing the Darboux vector of the path and its time derivative.

Musa K. Jouaneh [2], studied that Coordinating the tool (robot) and work piece (positioning table) motion in continuous manufacturing processes such as sealing, welding, and laser cutting offers several advantages over the motion of only the tool or work piece. Better utilization of the speed and workspace characteristics of the two devices, as well as improvement in tracking accuracy at sharp corners in the path, were some of the merits of that approach. In coordinated motion, the two devices were simultaneously moved to track a given path, and two strategies were developed for coordinating their motion, where each is dependent on the given path information. The first strategy was applicable to any type of path and resolves the motion by splitting the displacement between any two points on the path into segments moved by the robot and the table. The robot and the table move in opposite directions in this strategy. The second strategy was applicable to sharp cornered paths and resolves the given path into two smooth paths. The first path was a double clothoid curve, whereas the second path was a tangential straight-line path. The table executes a local motion around the corner in this strategy, whereas the robot is moved at all times. The uniqueness of this strategy in constructing the original corner path is proved. The developed strategies are shown to be applicable to other coordinated motion systems such as a robot on a mobile base and a robot with a rotary table.

Dongkyoung Chwa et.al [15], presented a novel approach to an online trajectory planning of robot arms for the interception of a fast maneuvering object under torque and velocity constraints. A body axis was newly introduced as a trajectory-planning coordinate in order to meet the position and the velocity matching conditions for a smooth grasp of the fast-maneuvering object. Using the position of the object and the end-effector in the inertia axis, the acceleration commands were generated in the X-, Y -, and Z-directions of the body axis and the acceleration commands were modified considering the torque and the velocity constraints. The trajectory planning in the X-direction became the speed planning to achieve the maximum speed, whereas the trajectory planning in the Y- and Z-directions became the direction planning where a missile-guidance algorithm was employed to intercept the maneuvering object. Finally, the acceleration commands in the body axis were transformed into the angle commands of the end-effector in the joint axis, which was used as the actual trajectory commands in robot arms.

Foudil [21], gave the generalized trajectory of a mobile Manipulator. The objective was to allow the end effector track a given trajectory in a fixed world frame. The motion of the platform and that of the manipulator were coordinated by a pseudo neural network designed from the kinematics model of the system. A learning paradigm was used to produce the required reference variables for each of the mobile platform and the robot manipulator for an overall coordinate behavior. Simulation results were presented to show the effectiveness of the proposed scheme.

Emmanuel A. Merchán et.al [17] presented a novel fuzzy genetic algorithm (GA) approach to tackling the problem of trajectory planning of two collaborative robot manipulators sharing a common workspace, where the manipulators had to consider each other as a moving obstacle whose trajectory or behavior was unknown and unpredictable, as each manipulator had individual goals and where both had the same priority. The goals were not restricted to a given set of joint values, but were specified in the workspace as coordinates at which it is desired to place the end-effector of the manipulator. By not constraining the goal to the joint space, the number of possible solutions that satisfies the goal increases according to the number of degrees of freedom of the manipulators. A

simple GA planner was used to produce an initial estimation of the movements of the robots' articulations and collision free motion was obtained by the corrective action of the collision-avoidance fuzzy units.

Kian Hsiang Low et.al [16], described a distributed layered architecture for resource-constrained multirobot cooperation, which was utilized in autonomic mobile sensor network coverage. In the upper layer, a dynamic task allocation scheme self-organized the robot coalitions to track efficiently across regions. It used concepts of ant behavior to self-regulate the regional distributions of robots in proportion to that of the moving targets to be tracked in a non-stationary environment. As a result, the adverse effects of task interference between robots were minimized and network coverage was improved. In the lower task execution layer, the robots used self-organizing neural networks to coordinate their target tracking within a region. Both layers employed self-organization techniques, which exhibit autonomic properties such as self-configuring, self-optimizing, self-healing, and self-protecting. Quantitative comparisons with tracking strategies such as static sensor placements, potential fields, and auction-based negotiation showed that our layered approach could provide better coverage, greater robustness to sensor failures, and greater flexibility to respond to environmental changes.

Alan Liu et.al [19] studied that in order to meet new mission requirements or to adapt environmental changes, robots perform a task switch or role switch. When a robot is required to perform a specific task, which is beyond its skills, it might need to request other robots' help. In order to improve the flexibility, they used the concept of a software mobile agent to design an architecture called Virtual Operator Multi Agent System (VOMAS). In VOMAS, intelligent agents, called a virtual Operator (VO) and a Robot Agent (RA), work together to control a robot to fulfill a specific task. A VO represented each task and the RA handles reactive control. Based on this architecture, VOMAS could perform the dynamic task switch to handle missions, which were not initially given. Two case studies, a wheelchair and formation control, were introduced for explaining our approach and to show the feasibility and effectiveness of dynamic task switch.

Furthermore, VOMAS could be treated as telerobotic architecture, and the network-load test showed that VOMAS requires less network load than direct control.

Xiaobu Yuan et.al [20] presented a novel approach of automated multi robot nano assembly planning. This approach used an improved self-organizing map to coordinate assembly tasks of nano robots while generating optimized motion paths at run time with a modified shunting neural network. It was capable of synchronizing multiple nano robots working simultaneously and efficiently on the assembly of swarms of objects in the presence of obstacles and environmental uncertainty. Operation of the presented approach was demonstrated with experiments at the end of the paper.

Mike Peasgood et.al [28] addressed the challenging problem of finding collision-free trajectories for many robots moving toward individual goals within a common environment. Most popular algorithms for multi-robot planning manage the complexity of the problem by planning trajectories for robots individually; such decoupled methods were not guaranteed to find a solution if one exists. In contrast, this paper described a multiphase approach to the planning problem that used a graph and spanning tree representation to create and maintain obstacle-free paths through the environment for each robot to reach its goal. The resulting algorithm guarantees a solution for a well-defined number of robots in a common environment. The computational cost was shown to be scalable with complexity linear in the number of the robots, and demonstrated by solving the planning problem for 100 robots, simulated in an underground mine environment, in less than 1.5 s with a 1.5 GHz processor. The practicality of the algorithm was demonstrated in a real-world application requiring coordinated motion planning of multiple physical robots.

Dong Sun et.al [10] studied that coordination of multi-robot systems has received extensive studies in the past decade. The majority of previous approaches required a complex setup of the hybrid position/force-control architecture, and has not fully addressed the coordination problem when the robots are not kinematically constrained but perform a common task. In this paper, they proposed to use a new coordination

scheme that was more straightforward and easier to implement and was applicable to a wider area. The basic idea of the new coordination strategy was to use the concept of motion synchronization, since the problem of coordinating multiple manipulators was basically the problem of maintaining certain kinematic relationships amongst robots. The key to the success of the new method was to ensure that each manipulator tracks its desired trajectory while synchronizing its motion with other manipulators' motions, so that differential (or synchronization) position errors amongst manipulators converge to zero. The controller, designed by incorporating the cross-coupling technology into adaptive-control architecture, successfully guarantees asymptotic convergence to zero of both position tracking and synchronization errors simultaneously. Experiments and simulations on multi-robot assembly systems demonstrated the effectiveness of the approach.

'Kyung-Seok Park et.al [18], The part of manipulators is normally studied with regularized environmental conditions. However, it is the most difficult that the part of AMR must be studied with uncertainty in the environmental conditions. The part of AMR has skeleton, sensor fusion, path planning etc. This paper is the research of the local pass planning that gathers information about external environment using neural network from each sensors and designs the algorithm which can determine which correct direction the robot can find. As the result of the research, AMR has been able to drive similarly as if the expert does and has been able to observe it acting without any control.

Xuena Qiu et.al [13], Motion planning with obstacles avoidance in uncertain environments was an essential issue in robotics. Complete coverage path planning of a mobile robot required the robot to pass through every area in the workspace with collision-free, which had many applications, e.g., various cleaning robots, painter robots, automated harvesters, land mine detectors and so on. A novel planning method integrating rolling windows and biologically inspired neural networks was proposed in this paper. The real-time environmental information could be represented by the dynamic activity landscape of the biological neural network. The rolling window approach was used to detect the local environments. Thus, a heuristic planning algorithm was

performed on-line in rolling strategy. Three cases on complete coverage path planning in uncertain environments and the comparison of the proposed method and simulations study the planning approach based on the biologically inspired neural networks. Simulation results showed that the proposed method was capable of planning collision free complete coverage robot motion path.

Meijuan Gao Jingwen Tian [26], A mobile robot path planning method based on improved simulated annealing algorithm and artificial neural network was proposed. First the simulated annealing algorithm with the best reserve mechanism was introduced and it was combined with Powell algorithm to form improved simulated annealing mixed optimize algorithm which not only added the good solution protective measures but also improved the convergence rate of simulated annealing algorithm. Then we took the obstacle collision penalty function which was expressed using neural network and the path length as the energy function of improved simulated annealing mixed optimize algorithm, thereby the solution which obtained by the improved simulated annealing mixed optimize algorithm could not only satisfy the path shortest but also effective avoid the collision with obstacle. The simulation result showed that the proposed method was feasible and valid.

Hairong Xiao Li Liao et.al [22] studied that Path planning is a difficult part of the navigation task for the mobile robot under dynamic and unknown environment. It was needed to solve a mapping relationship between the sensing space and the action space. The relationship could be achieved through different ways. But it was difficult to be expressed by an accurate equation. This paper used multi-layer feed forward Artificial Neural Network (ANN) to construct a path-planning controller by its powerful nonlinear functional approximation. Then the path planning task was simplified to a classified problem which are five state-action mapping relationship. One reinforcement learning method, Q-learning, was used to collect training samples for the ANN controller. At last the trained controller runs in the simulation environment and retrained it furthermore combining the reinforcement signal during the interaction with the environment. Strategy based on the Combination of ANN and Q-learning, Q-ANN, was better than using only

one of the two methods. The simulation result also showed that the strategy could find the optimal path than using Q learning only.

Xiaobu Yuan et.al [24] have seen that a modification to the well-known shunting equation has successfully created a method of automated path generation. They further investigated the integration of automated path generation with task allocation in multi-robot coordination. It presented an algorithm that was able to equally distribute the workload of multiple mobile robots while guiding them to move along optimized paths in the presence of obstacles and environmental uncertainty. Experiment results were also provided in the paper to examine the operation of the proposed algorithm and its application.

Huan Tan et.al [25], studied that in the human-robot mixed and obstacle existing environment, several robots can cooperate to complete a single task or a group of tasks, which had the close connections. In this paper they proposed an intelligent algorithm, which was based on the statistics method and artificial neural networks for the multiple robots cooperation in this environment. This algorithm mainly involved the algorithm for cooperation in the layer of task and the motion track planning. Some simulation experiments had been carried to valid the algorithm and they got satisfying experimental results, which demonstrated the effectiveness and advantage.

Ladaloharivola Randria et.al [23] found that the optimal path search was important in navigation learning with an electrical wheelchair. In this paper they studied six basic approaches for path planning in a static environment: Breath-First Search, Depth-First Search, A*, Moore-Dijkstra, Neural approach and genetic algorithms. Firstly, the environment was modeled in a 2D grid and the genetic approach had fixed chromosome length. Then the genetic approach was evaluated within a grid free environment and with variable chromosome lengths. The latter approach offered good solution in precision and in computation time, under certain conditions. Tests were led and the results were discussed through experiments.

Zeungnam Bien et.al [3], proposed a trajectory planning method that ensured collision-free and time-optimal motions for two robotic manipulators with limited actuator torques and velocities. In terms of parameters defining the positions along the paths of two robots, a two-dimensional coordination space and a maximum velocity curve were constructed to detect collision regions and to plan a time-optimal velocity curve, respectively. An algorithm was then provided for the case of zero initial velocities and a single collision region.

Mitsuo Gent et.al [4], in this study, they investigated the possibility of using genetic algorithms to solve shortest path problems. The most thorny and critical task for developing a genetic algorithm to this problem was how to encode a path in a graph into a chromosome. A priority-based encoding method was proposed which could potentially represent all possible paths in a graph. Because a variety of network optimization problems may be solved, either exactly or approximately, by identifying shortest path, this study provided a base for constructing efficient solution procedures for shortest path based network optimization problems. The proposed approach has been tested on three randomly generated problems with different size from 6 nodes to 70 nodes and from 10 edges to 211 edges. The experiment results were very encouraging: it could find the known optimum very rapidly with very high probability. It can be believed that genetic algorithms may hopefully be a new approach for such kinds of difficult-to-solve problems.

Shuai Li et.al [27], in this paper, a neural network approach named shortest path neural networks (SP-NN) was proposed for real-time on-line path planning. Based on grid-based map and mapping this kind of map to neural networks, this proposed method was capable of generating the globally shortest path from the target position to the start position without collision with any obstacles. The dynamics of each neuron was distinctive to other previously presented methods by other researchers and ensures that the generated path was shortest without collision and that the state of neurons varied continuously. Extensive simulations showed the efficiency of the presented method.

# Chapter 1

# Introduction

Robot is derived from a Czech word meaning "menial labor". A robot is a mechanical or virtual, artificial agent. It is usually an electromechanical system, which, by its appearance or movements, conveys a sense that it has intent or agency of its own. They are capable of performing many different tasks and operations precisely and do not require any common safety and comfort elements. They posses a number of links attached serially to each otherwith joints, where each joint can be moved by some type of actuator.

Today, robots are used in many ways, from lawn mowing to auto manufacturing. Scientists see practical uses for robots in performing socially undesirable, hazardous or even "impossible" tasks --- trash collection, toxic waste clean up, desert and space exploration, and more. AI researchers are also interested in robots as a way to understand human (and not just human) intelligence in its primary function -- interacting with the real world.

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The neural network architecture and training techniques are used to organize the movements of robots to the desired location. To train the robots, self-organizing feature map characteristics of Artificial Neural Network have been used. In which, Kohonen Learning Rule has been used.

## 1.1 Robotics

Robotics is the science and technology of robots, their design, manufacture, and application. Robotics requires a working knowledge of electronics, mechanics and software, and is usually accompanied by a large working knowledge of many subjects. A person working in the field is a roboticist. Robotics systems consist of not just robots, but

also other devices and systems that are used together with the robots to perform the necessary tasks.Robotics is an inter disciplinary subject that benefits from mechanical engginerring, electrical and electronic enggnerring, computer science, biology, and many other disiciplenes.Robotics are used in industrial, military, exploration, home making, and academic and research applications.

## 1.2 Robot Components

A Robotic system consists of various elements like:

**1.2.1 Manipulator**:- This is the main body of Robot and consists of the links, the joints, and other structural elements of the Robot. The robot manipulator can be divided into two sections, each with a different function:

(i) Arm and Body - The arm and body of a robot are used to move and position parts or tools within a work envelope. They are formed from three joints connected by large links.

(ii) Wrist - The wrist is used to orient the parts or tools at the work location. It consists of two or three compact joints.

**1.2.2 End-effector**: - In robotics, an end effector is a device or tool connected to the end of a robot arm. The structure of an end effector, and the nature of the programming and hardware that drives it, depends on the intended task. .

**1.2.3 Actuators**: - The actuators are the 'muscles' of a robot; the parts which convert stored energy into movement. By far the most popular actuators are electric motors, but there are many others, some of which are powered by electricity, while others use chemicals, or compressed air.Eg.are Motors, Stepper Motors, Piezo Motors, Air Muscles, Electroactive Polymers

**1.2.4 Sensors: -**Sensors are used to collect information about the internal state of the robot or to communicate with the outside world .As in humans the Robot controllers needs to know where each link of the robot is in order to know the robot's configuration. Choosing a sensor for a particular application depends on various characteristics like cost of the sensor, size and weight of the sensor, output of the sensor that may be digital or

analog, how a sensor can interface with other devices, its resolution, reliability, accuracy, sensitivity, linearity etc.

Sensors are broadly classified as:
(i) Position sensors: Used to measure displacements, both rotary and linear, as well as movements.
(ii) Velocity sensors: Very much similar to position sensors, can measure rotational and linear velocity.
(iii) Force and pressure sensors: They are used for dynamic purposes and measure forces.

**1.2.5 Controllers**: - The controller is a major component of a robot, which directs the end effector to move in a desired sequence, and to pass through desired points. It also functions to store position and sequence data in memory so that program can be repeated. Robot controllers range in complexity from simple stepping switches through pneumatic logic sequencers, diode matrix boards, electronic sequencers, and microprocessor to minicomputers.

**1.2.6 Processor**: - The processor is the brain of the robot. It calculates the motions of the robot's joints, determine how much and how fast each joint must move to achieve the desired location and speed, and oversees the coordinated actions of the controller and the sensors. The processor is generally a computer, which works like all other computers, but is dedicated to a single purpose. It requires an operating system, programs, and peripheral equipment such as monitors and has many of the same limitations and capabilities of a PC processor.

**1.2.7 Software**: - There are perhaps three groups of software that are used in a robot. One is the operating system, which operate the computer. The second is the robotic software, which calculates the necessary motions of each joint based on the kinematic equation of the robot. The third group is the collection of routines and application programs that are developed in order to use the peripheral devices of the robots, such as vision routines, or to perform specific tasks.

## 1.3 Degrees of Freedom

In Robotics, degrees of freedom (DOF) are the set of independent displacements and/or rotations that specify completely the displaced or deformed position and orientation of the body or system.The number of DOF that a manipulator possesses is the number of independent position variables that would have to be specified in order to locate all parts of the mechanism. In other words, it refers to the number of different ways in which a robot arm can move. In the case of typical industrial robots, because a manipulator is usually an open kinematic chain, and because each joint position is usually defined with a single variable, the number of joints equals the number of degrees of freedom.

In total, our arm has seven degrees of freedom: three in the shoulder, one in the elbow, and three in the arm below the elbow. Three degrees of freedom are sufficient to bring the end of a robot arm to any point within its workspace, or work envelope, in three dimensions. Thus, in theory, a robot should never need more than three degrees of freedom. But the extra possible motions, provided by multiple joints, give a robot arm versatility that it could not have with just three degrees of freedom.

## 1.4 Robot Joints and Coordinates

Most robots have either a Prismatic joint or a Revolute joint.

(i) Prismatic joints are linear; there is no rotation involved. They are either hydraulic or pneumatic cylinders or, they are linear electric actuators. These joints are used in gantry, cylindrical or, similar joint configuration.

(ii) Revolute joints are rotary, and although hydraulic and pneumatic rotary joints are common, most rotary joints are electrically driven, either by stepper motors or, more commonly by servomotors.

Robot configurations generally follow the coordinate frames with which they are defined. The following configurations are common.

(i) Cartesian: - These robots are made of three linear joints that position the end effector, which are usually followed by additional revolute joint that orientate the end effector.

(ii) Cylindrical: - They have two prismatic joints and one revolute joint for positioning the part, plus revolute joints for orienting the part.
(iii) Spherical: - They follow a spherical coordinate system, which has one prismatic and two revolute joints for positioning the part, plus additional revolute joints for orientation.
(iv) Articulated: - They are all revolute, similar to a human arm. They are perhaps the most common configuration for industrial robots.
(v) Selective Compliance Assembly Robot Arm (SCARA): - They have two revolute joints that are parallel and allow the robot to move in horizontal plane, plus an additional prismatic joint that moves vertically.

## 1.5 Robot Programming Modes

Robots may be programmed in number of different modes, depending on the robot. The following programming modes are common.
(i) Physical setup: - In this mode, an operator sets up switches and hard stops that control the motion of the robot. This mode is usually used along with other devices, such as Programmable Logic Controllers (PLC).
(ii) Teach mode: - Here robot joints are moved with teach pendant. When the desired location is achieved, the location is entered into the controller. During Play back, the controller will move the joints to the same locations and orientations.
(iii) Continuous Walk Trough Mode: - All robot joints are moved simultaneously, while the motion is continuously sampled and recorded by the controller. During Playback, The exact motion that was recorded is executed.
(iv) Software mode: - A program is written offline or online and is executed by the controller to control the motions. This mode is sophisticated and versatile mode and can include sensory information, conditional statements and branching.

## 1.6 Advantages

a) Increase Productivity with Shorter Cycle Time
b) Increase Quality of Product, Process and Work Environment
c) Increase Manufacturing Flexibility
d) Reduce Scrap and Manufacturing Costs

e) Compete Better By Reducing Undesirable Tasks
f) Improved Worker Safety
g) Decrease Floor Space
h) GUI Setup for Fast & Easy Operation
i) Cluster Machining in One Cell
j) Automatic Changeover for Different Products
k) Re-Program Equipment for Different Process
l) Stabilizes Production

## 1.7 Disadvantages

a) High initial cost
b) Need for extra space, and new technology
c) Need highly skilled and technical engineers, programmers
d) Possible injuries during working with robot
e) Limited functions
f) New systems bring out defects
g) Intellectual or physical limitations of employees
h) Replace certain workers causing economic losses
i) Certain military robots are pro-war
j) Making companies and human beings more dependent

## 1.8 Artificial Neural Networks

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones.

### 1.8.1 Training and Learning

The method of setting the value for the weights enables the process of learning or training. The process of modifying the weights in the connections between network layers with the objective of achieving the expected output is called training a network. The internal process that takes place when a network is trained is called learning.

Training for adaptive neural networks can be classified into three major categories:
(i) Supervised Training, which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised training include error-correction learning, reinforcement learning and stochastic learning.
(ii) Unsupervised Training uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised training are Hebbian learning and competitive learning.
(iii) In this method, a teacher is also assumed to be present, but the right answer is not presented to the network. Only an indication of whether the output answer is right or wrong. The network must use this information to improve its performance. Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

Various learning rules used in a neural network are given below
(i) Hebbian Learning is the oldest method. It states that "when an axon of cell A is near enough to excite cell B and repeatedlly or persistently takes part in firing it, some growth process or metallic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is incresead".
(ii) Perceptron learning rule is an example of supervised training. is an input to the network and is the corresponding target output. As each input is applied to the network, the network output is compared to the target. The learning rule then adjusts the weights and biases of the network in order to move the network output closer to the target.

(iii) Delta learning is one of the most commonly used learning rules. For a given input vector, the output vector is compared to the correct answer. If the difference is zero, no learning takes place; otherwise, the weights are adjusted to reduce this difference.

(iv) Competitive learning is a rule based on the idea that only one neuron from a given iteration in a given layer will fire at a time. Weights are adjusted such that only one neuron in a layer.

(v) Out star learning can be explained when the neiurons are arranged in a layer.this rule is designed to produce the desired response from the layer of neurons.the rule is to provide learning of repetitive and charchtertic properties of input output relationships.

(vi) Boltzmann learning is stochastic learning. In this learing neurons constitute a recurrent stucure and they work in binary form. This leraning is charcterized by an energy function, E,the value of which is determinde by the particular states occupied by the individual neurons.

## 1.9 Types of Artificial Neural Network

There are various types of neural networks, some are given below:

### 1.9.1 Feed-forward network

Feed-forward ANN allows signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANN tends to be straightforward networks that associate inputs with outputs. It is extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

### 1.9.2 Feedback network

Feedback network can have signals traveling in both directions by introducing loops in the network. Feedback network is very powerful and can get extremely complicated. Feedback network is dynamic; its 'state' is changing continuously until they reach an equilibrium point. It remains at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architecture is also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations.

### 1.9.3 Perceptron Network

It consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. In this way it can be considered the simplest kind of feed-forward network. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1).

### 1.9.4 Adaline Network

This network uses Least Mean Sqaure error learning rule and is applied to various neural network applications.It uses bipolar activations for its input signals and target output. The weights and the bias of the adaline are adjustable.The initial weights of adaline network have to be set to small random values and not to zero. After that the activations for the input unit are set. The net input is calculaed based on the trainging input patterns and weights. The training process is continued until the error difference between target and the net input becomes minimum.

### 1.9.5 Madaline Network

Madaline is the combination of adalines.It is also called multilayered adaline. If the adalines are combined, such that the output of some of them becomes input of others, then the net becomes multilayered. Madaline has two training algorithm, MRI and MRII.
In MRI, the weights of the hidden adaline units are only adjusted while the weights of the output units are fixed.In MRII, all weights present in the net are updated. Thus it allows traing of weights in all layers of net. The total error of any input pattern is given as the sum of squares of the errors at each output unit.Thie algorithm is different from MRI algorithm in the manner of the weight updation only.

### 1.9.6 Associative Memory Networks

Pattern Association is the process of forming associtions between related patterns. The patterns may be of same type or of a different type.Associate neural Nets are single layer nets in which the weights are detrmined to store an asset of pattern associations.The

associative nets mainly consists of two types of nets. If the input vector pair is same as the output vector pair, then it results in an auto associative net. If the input vector pair is different from that of the output vector pair, then it forms a hetero associative net.

## 1.10 Self Oraganizing Map

Self-Organizing Map (SOM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two dimensional), discretized representation of the input space of the training samples, called a map. The map seeks to preserve the topological properties of the input space. The model was first described as an artificial neural network by the Finnish professor Teuvo Kohonen, and is sometimes called a Kohonen map.

Training Algorithm is as follows:

Step 1: Each node's weights are initialized.

Step 2: A vector is chosen at random from the set of training data and presented to the lattice.

Step 3: Every node is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).

Step 4: The radius of the neighborhood of the BMU is now calculated. This is a value that starts large, typically set to the 'radius' of the lattice, but diminishes each time-step. Any nodes found within this radius are deemed to be inside the BMU's neighborhood.

Step 5: Each neighboring node's (the nodes found in step 4) weights are adjusted to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered.

Step 6: Repeat step 2 for N iterations.

# Chapter 2

# Multi Mobile Robot Modeling

## 2.1 Multi Robot Systems

Today Robot Systems are becoming more and more significant in various aspects of human life, for example in industrial, commercial and scientific applications. As a result of scientific achievements and industrial development, the number of robots currently being used in industrial projects is increasing fast. However, robotics has been evolving in the last years and there has been an increasing interest in developing Multi-robot systems, capable of performing robust cooperative work. Various robots work in same workspace, they work under same condition, under same environment. They can work for same target and also can for different targets.

Figure 2.1: Central coordinated multi robots

Properties of Multi Robot System are:

I. Explorations in hazardous environments
II. Execution of a task beyond the limits of single robots
III. Ability to complete a task more rapidly
IV. Performance of a complex task by multiple specialized robots (rather than 1 superbot)
V. Highly distributed sensing (sensor nets) with mobile sensors
VI. Gives an economic system

## 2.2 Mobile Robots

A Mobile Robot is an automatic machine that is capable of movement in a given environment. Mobile robots have the capability to move around in their environment and are not fixed to one physical location. In contrast, industrial robots usually consist of a jointed arm (multi-linked manipulator) and gripper assembly (or end effector) that is attached to a fixed surface.

There are many types of mobile robot navigation:

(i) Manual remote: - A manually tele-op'd robot is totally under control of a driver with a joystick or other control device. The device may be plugged directly into the robot, may be a wireless joystick, or may be an accessory to a wireless computer or other controller. A tele-op'd robot is typically used to keep the operator out of harm's way. Examples of manual remote robots include Foster-Miller's Talon, iRobot's PackBot and KumoTek's MK-705 Roosterbot.

(ii) Guarded: - A guarded tele-op robot has the ability to sense and avoid obstacles but will otherwise navigate as driven, like a robot under manual tele-op. Few if any mobile robots offer only guarded.

(iii) Line-following robot: - Some of the earliest Automated Guided Vehicles (AGVs) were line following mobile robots. They might follow a visual line painted or embedded in the floor or ceiling or an electrical wire in the floor. Most of these robots operated a simple "keep the line in the center sensor" algorithm. They could not circumnavigate obstacles; they just stopped and waited when something blocked their path. Many examples of such vehicles are still sold, by Transbotics, FMC, Egemin, HK Systems and many other companies.

(iv) Autonomously randomized robot: - Autonomous robots with random motion basically bounce off walls, whether those walls are sensed with physical bumpers like the Roomba cleaners or with electronic sensors like the Friendly Robotics lawn mower. The simple algorithm of bump and turn 30 degrees leads eventually to coverage of most or all of a floor or yard surface.

(v) Autonomously guided robot: - An autonomously guided robot knows at least some information about where it is and how to reach various goals and or waypoints along the way. "Localization" or knowledge of its current location, is calculated by one or more

means, using sensors such motor encoders, vision, Stereopsis, lasers and global positioning systems. Positioning systems often use triangulation, relative position and/or Monte-Carlo/Markov localization to determine the location and orientation of the platform, from which it can plan a path to its next waypoint or goal. It can gather sensor readings that are time- and location-stamped, so that a hospital, for instance, can know exactly when and where radiation levels exceeded permissible levels. Such robots are often part of the wireless enterprise network, interfaced with other sensing and control systems in the building. For instance, the PatrolBot security robot responds to alarms, operates elevators and notifies the command center an incident arises. Other autonomously guided robots include the SpeciMinder and the Tug delivery robots for hospital labs, though the latter actually has people at the ready to drive the robot remotely when its autonomy fails. The Tug sends a letter to its tech support person, who then takes the helm and steers it over the Internet by looking through a camera low in the base of the robot.

(vi) Sliding autonomy: - More capable robots combine multiple levels of navigation under a system called sliding autonomy. Most autonomously guided robots, such as the HelpMate hospital robot, also offer a manual mode. The MOBILE-ROBOTS inside guidance system, which is used in the ADAM, PatrolBot, Speci-Minder, MapperBot and a number of other robots, offers full sliding autonomy, from manual to guarded to autonomous modes.


Figure 2.2: Examples of mobile robots

## 2.3 Robot Kinematics

In order to adjust the robot's controllers, a kinematic and dynamic mathematical model of the robot (include the robot's physical characteristics) are needed. The kinematics is the study of the robot's movements with regard to a reference system. It is an analytic description of the spacial movement of the robot like a function of time and a relationship, between the position and the orientation of the robot's external link and the values of their joint co-ordinates.

Robot kinematics are mainly of the following two types

(i) Forward kinematics

(ii) Inverse kinematics.

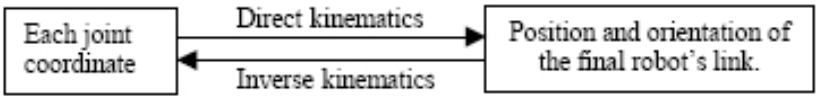


Figure 2.3: Forward and Inverse kinematics

### 2.3.1 Forward Kinematics

In Forward Kinematics, Each Joint angle is given,the position and orientation of end effector is calculated.

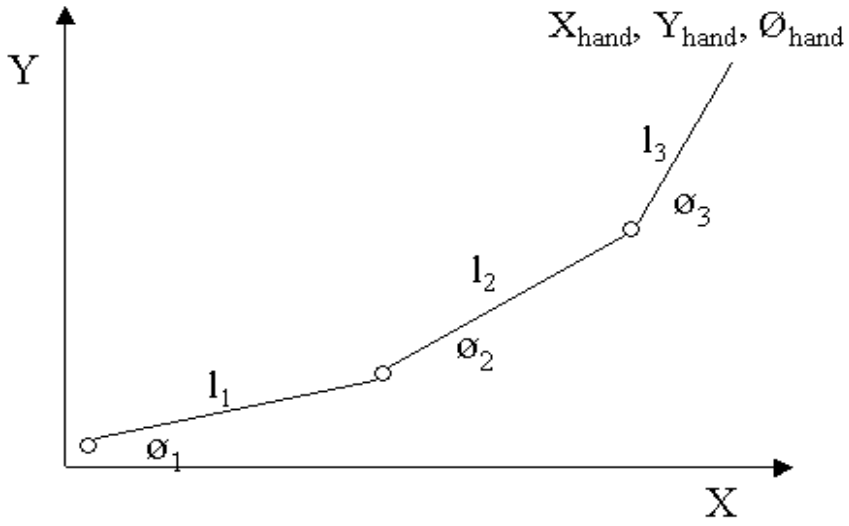


Figure 2.4: Forward kinematic plane

It is also known as direct kinematics. In forward kinematics, the length of each link and the angle of each joint is given and we have to calculate the position of any point in the work volume of the robot. As shown above in Figure 2.4

(i) Forward position kinematics

The forward position kinematics (FPK) solves the following problem: "Given the joint positions, what is the corresponding end effector's pose?". One given joint position vector always corresponds to only one single end effector pose. The FK problem is not difficult to solve, even for a completely arbitrary kinematic structure.

In this following possibilites are considered:

(a) Cartesian (gantry, rectangular) coordinates.

(b) Cylinderical coordinates.

(c) Spherical coordinates.

(d) Articulated coordiantes.

(ii) Forward velocity kinematics

The forward velocity kinematics (FVK) solves the following problem: "Given the vectors of joint positions and joint velocities, what is the resulting end effector twist?". One given set of joint positions and joint velocities always corresponds to only one single end effector twist. Its possible configurations are:

(a) Roll, Pitch, Yaw

(b) Euler Angles

(c) Articulated joints

(iii) Forward force kinematics

The forward force kinematics (FFK) solves the following problem: "Given the vectors of joint force/torques, what is the resulting static wrench that the end effector exerts on the environment?"

### 2.3.2 Inverse Kinematics

In inverse kinematics, the length of each link and position of the point in work volume is given and we have to calculate the angle of each joint.As shown Below:

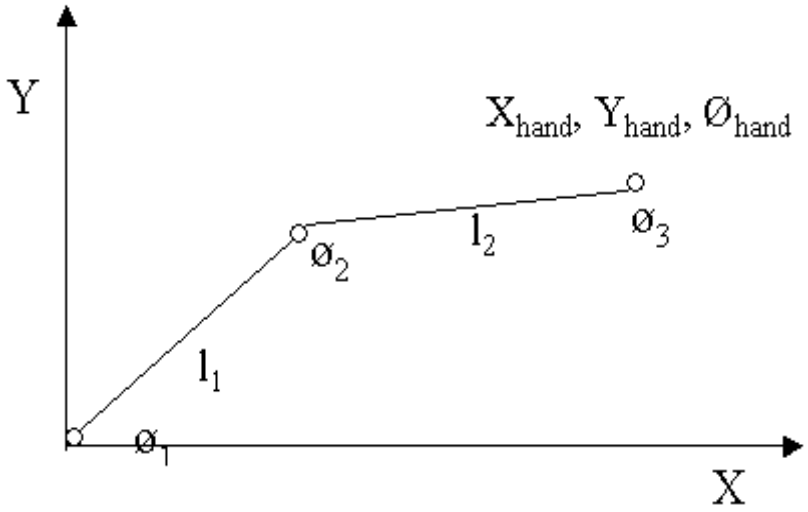


Figure 2.5: Inverse kinematic plane

(i) Inverse Position Kinematics

The Inverse Position Kinematics (IPK) solves the following problem: "Given the actual end effector pose, what are the corresponding joint positions?" In contrast to the forward problem, the solution of the inverse problem is not always unique: the same end effector pose can be reached in several configurations, corresponding to distinct joint position vectors. Its possibiltes are same as in the case of Forward Kinematics

(ii) Inverse Velocity Kinematics

Assuming that the Inverse Position Kinematics problem has been solved for the current end effector pose, the Inverse Velocity Kinematics (IVK) then solves the following problem: "Given the end effector twist, what is the corresponding vector of joint velocities?". Again, configuratios are same.

(iii) Inverse Force Kinematics

Assuming that the Inverse Position Kinematics problem has been solved for the current end effector pose, the Inverse Force Kinematics (IFK) then solves the following problem: "Given the wrench that acts on the end effector, what is the corresponding vector of joint forces/torques?"

### 2.3.3 D-H Representation of Forward Kinematic Equations

The D-H method allows the step from a link to the following link by 4 basic transformations that depends only on the robot's constructive characteristics.

These are basic transformations that relate the reference system of the element n+1 with the reference system of the element n.

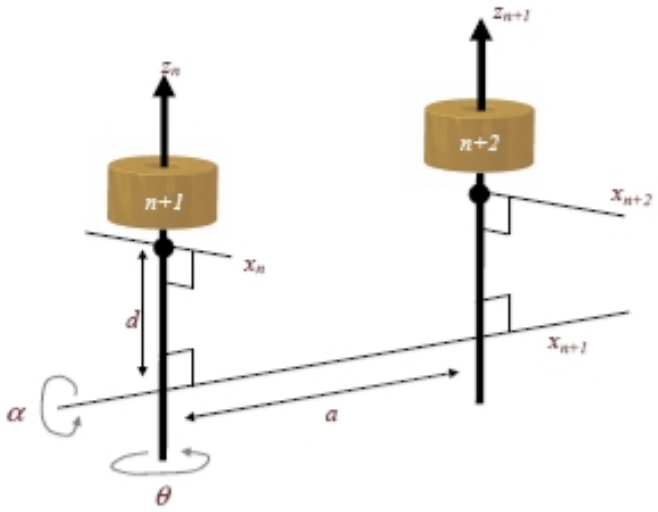


Figure 2.6: D-H representation of general joint link combination

1. A rotation θn+i about the Zn axis (to bring Xn parallel with Xn+1)
2. A translation dn+i along the Zn axis (to make the x-axes collinear)
3. A translation an+i along the X-axis (to make the z-axes coincide)
4. A rotation αn+i about the Xn axis (to bring Zn parallel with Zn+1)

## 2.4 Trajectory Planning

Path and trajectory planning relates to the way a robot is moved from one location to another in a controlled manner. A path is defined as a sequence of robot configurations in a particular order without regard to the timing of these configurations.However, a trajectory is concerned about when each part of the path must be attained, thus

specifying timing. A tarjectory depends upon the velocities and the accelerations at different times where as path is independent of it.

The trajectory planning can be done in two ways:

(i) Joint Space Trajectory

(ii) Cartesian Space Trajectory

### 2.4.1 Joint Space

A trajectory planning approach for controlling flexible robots is described here. It is demonstrated that choosing actual joint angles as the generalized rigid coordinates is the key to applying the proposed approach. From the observation of the special structure of the input matrix, the concepts of motion-induced vibration and inverse dynamics under a specified motion history of the joints are formed naturally. Based on the above concepts, trajectory planning in joint space is proposed by using the optimization technique to determine the motion of joints along a specified path in joint space or workspace and for general point-to-point motion. The motion for each joint is assumed to be in a class consisting of a fifth-order polynomial and finite terms of Fourier series. This parameterization of motion allows the optimal trajectory planning to be formulated as a standard nonlinear programming problem, which avoids the necessity of solving a two-point-boundary-value problem and using dynamic programming. Setting the accelerations to zero at the initial and the final times is used to obtain smoother motion to reduce the spillover energy into unmodeled high-frequency dynamics.

### 2.4.2 Cartesian Space

The Robot is to track a specified path for the purpose of accomplishing the task satisfactorily and avoiding collisions with obstacles in the workspace. In reality, typical robotic tasks include moving the end-effector to specified Cartesian positions or along a specified Cartesian path. Continuous- Path (CP) robotics applications appear in operations such as arc welding, flame cutting, and routing etc. Robot trajectory planning refers to the development of time history of position, velocity, and acceleration for each degree of freedom. To achieve a smooth motion, not only the position but also the velocity must be prescribed at every intermediate point along the motion path. Trajectory

planning can be conducted either in the joint-variable space or in the Cartesian space. However, for trajectory planning in Cartesian space, the transformation from Cartesian to joint coordinates in real-time is required. As the control of robot motion is carried out at the joint level, the computational complexities involved in trajectory planning and ordinate transformation have hindered the on-line implementation of Cartesian based path planning. The path is divided into m segments. It was assumed that the traveling path could be specified by a group of parameter equations in Cartesian coordinates.

The method described here has the merits as follows:

(1) The co-efficient of the polynomial of each segment of the path are obtained in recursive form, so the algorithm can be easily converted into computer programs

(2) The method needs less computation and obtains approximate minimum-time trajectories.

# Chapter 3

# Artificial Neural Networks

## 3.1 Introduction to Neural Networks

The term neural network is used to refer a network or circuit of biological neurons. It is an interconnected assembly of simple processing elements,units or nodes.The processing ability of the network is stored in the interunit connection strengths,or weights,obtained by process of adation to or learning from,a set of training patterns.

The neuron has four main regions to its structure. The cell body, or soma, has two offshoots from it, the dendrites, and the axon, which end in presynaptic terminals. The cell body is the heart of the cell, containing the nucleus and maintaining protein synthesis. A neuron may have many dendrites, which branch out in a treelike structure, and receive signals from other neurons. A neuron usually only has one axon which grows out from a part of the cell body called the axon hillock. The axon conducts electric signals generated at the axon hillock down its length. These electric signals are called action potentials. The other end of the axon may split into several branches, which end in a presynaptic terminal. Action potentials are the electric signals that neurons use to convey information to the brain. All these signals are identical. Therefore, the brain determines what type of information is being received based on the path that the signal took. The brain analyzes the patterns of signals being sent and from that information it can interpret the type of information being received.

Neural networks are a form of multiprocessor computer system, with

- Simple processing elements
- A high degree of interconnection
- Simple scalar messages
- Adaptive interaction between elements

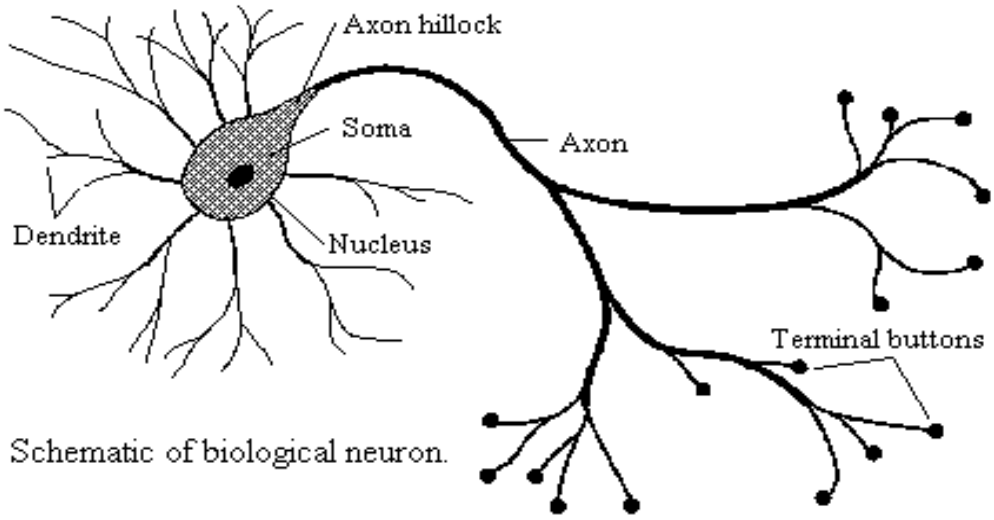


Figure 3.1: A biological neuron

### 3.1.1 Historical Background

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras.

Many important advances have been boosted by the use of inexpensive computer emulations. Following an initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, relatively few researchers made important advances. These pioneers were able to develop convincing technology, which surpassed the limitations identified by Minsky and Papert. Minsky and Papert, published a book (in 1969) in which they summed up a general feeling of frustration (against neural networks) among researchers, and was thus accepted by most without further analysis. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding.

The first artificial neuron was produced in 1943 by the neuro-physiologist Warren McCulloch and the logician Walter Pits. But the technology available at that time did not allow them to do too much.

## 3.2 Introduction to Artificial Neural Networks

Artificial neural networks, consists of an interconnected group of artificial neurons or nodes. It is inspired by the way biological nervous systems, such as brain,process information. The term Neural Network and Artificial Neural Network has two distinct usages:

Biological neural networks are made up of real biological neurons that are connected or functionally-related in the peripheral nervous system or the central nervous system. In the field of neuroscience, they are often identified as groups of neurons that perform a specific physiological function in laboratory analysis.

Artificial neural networks are made up of interconnecting artificial neurons (programming constructs that mimic the properties of biological neurons). Artificial neural networks may either be used to gain an understanding of biological neural networks, or for solving artificial intelligence problems without necessarily creating a model of a real biological system.

### 3.2.1 Basic Architecture of Artificial Neural Network

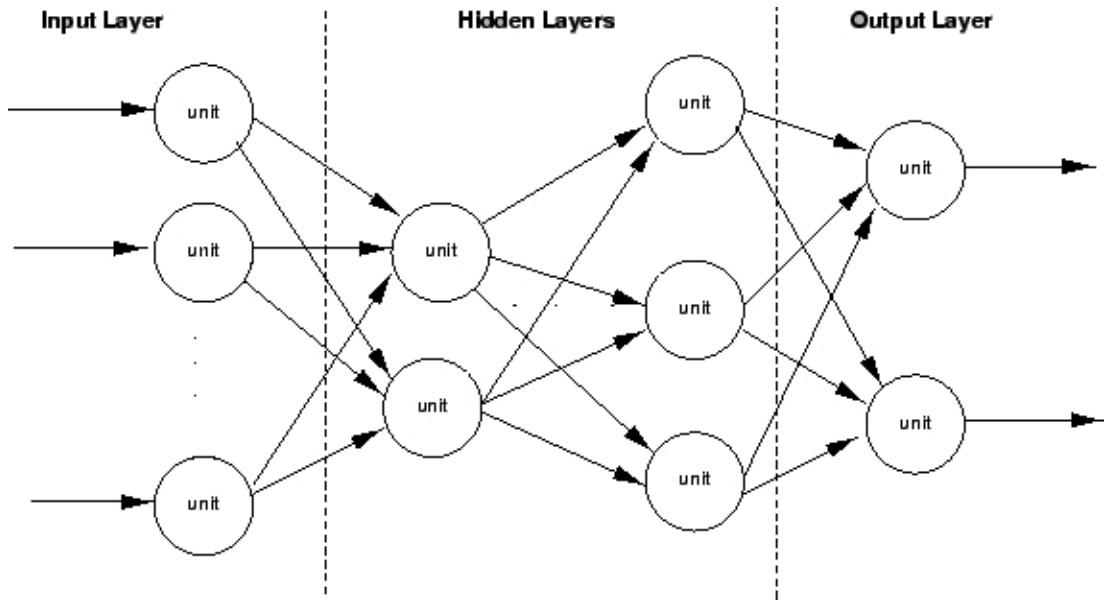


Figure 3.2: Architecture of artificial neural network

The arrangement of neurons into layers and the pattern of connection within and in between layer are genrally called as the architecture of the net.

A typical Artificial Neural Netwok, abbreviated as ANN,comprises an input layer, output layer and hidden(intermediate) layer of neurons. ANNs are often called layered networks. A three layer ANN is shown in Figure 3.2

The activity of the input units represents the raw information that is fed into the network.

The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.

The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

## 3.2.2 Artificial Neural Network Terminologies

The key terms that we will use through out this chapter are discussed bellow:

**3.2.2.1 Nodes:** In a neural network model simple nodes, which can be called variously "neurons", "neurodes", "units", are connected together to form a network of nodes. They are the elements to which input and output has be connected.

**3.2.2.2 Weights:** Weight is an information used by a neural net to solve a problem.During learning and training process these weights are updated.

**3.2.2.3 Firing Rule:** A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained.

**3.2.2.4 Transfer Function:** It is an input-output function specified for the units. This function typically falls into one of three categories:

For linear units, the output activity is proportional to the total weighted output.

For threshold units**,** the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurones than do linear or threshold units.

**3.2.2.5 Bias:** A bias acts exactly as a weight on a connection from a unit whose activation is always 1.

**3.2.2.6 Threshold:** The threshold 'θ' is a factor which is used in calculating the activations of the given net. User defines the threshold value.

**3.2.2.7 Learning Rules:** A Neural Network learns about its environment through an interactive process of adjustments applied to its synaptic weights and bias level.

### 3.2.3 Advantages of Artificial Neural Networks

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
- Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

The main disadvantage of ANN is they operate as black boxes. The rules of operation in neural networks are completely unknown. It is not possible to convert the neural structure into known model structures such as ARMAX, etc. Another disadvantage is the amount of time taken to train networks. It can take considerable time to train an ANN for certain functions.

## 3.3Training of Artificial Neural Networks

A neural network has to be configured such that the application of a set of inputs produces (either 'direct' or via a relaxation process) the desired set of outputs. Various

methods to set the strengths of the connections exist. One way is to set the weights explicitly, using a prior knowledge. Another way is to 'train' the neural network by feeding it teaching patterns and letting it change its weights according to some learning rule. We can categories the learning situations in three distinct sorts. These are supervised learning, unsupervised learning and reinforcement learning. Usually any given type of network architecture can be employed in any of those tasks.

### 3.3.1 Supervised learning

In supervised learning, we are given a set of example pairs (x,y), $x \in X$, $y \in Y$ and the aim is to find a function f in the allowed class of functions that matches the examples. In other words, we wish to infer the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error which tries to minimize the average error between the network's output, f(x), and the target value y over all the example pairs. When one tries to minimise this cost using gradient descent for the class of neural networks called Multi-Layer Perceptrons, one obtains the common and well-known backpropagation algorithm for training neural networks.

Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition). This can be thought of as learning with a "teacher," in the form of a function that provides continuous feedback on the quality of solutions obtained thus far.

### 3.3.2 Unsupervised learning

In unsupervised learning we are given some data x, and the cost function to be minimized can be any function of the data x and the network's output, f.

The cost function is dependent on the task (what we are trying to model) and our a priori assumptions (the implicit properties of our model, its parameters and the observed variables).

As a trivial example, consider the model f(x) = a, where a is a constant and the cost C = $E[(x - f(x))^2]$. Minimizing this cost will give us a value of a that is equal to the mean of the data. The cost function can be much more complicated. Its form depends on the application: For example in compression it could be related to the mutual information between x and y. In statistical modelling, it could be related to the posterior probability of the model given the data. (Note that in both of those examples those quantities would be maximized rather than minimised).

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

### 3.3.3 Reinforcement learning

In reinforcement learning, data x is usually not given, but generated by an agent's interactions with the environment. At each point in time t, the agent performs an action $y_t$ and the environment generates an observation $x_t$ and an instantaneous cost $c_t$, according to some (usually unknown) dynamics. The aim is to discover a policy for selecting actions that minimizes some measure of a long-term cost, i.e. the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated.

More formally, the environment is modeled as a Markov decision process (MDP) with states $s_{1,\ldots\ldots\ldots}s_n \in S$ and actions $a_1,\ldots\ldots..a_n$ with the following probability distributions: the instantaneous cost distribution $P(c_t \mid s_t)$, the observation distribution $P(x_t \mid s_t)$ and the transition $P(s_{t+1} \mid s_t, a_t)$, while a policy is defined as conditional distribution over actions given the observations. Taken together, the two define a Markov chain (MC). The aim is to discover the policy that minimizes the cost, i.e. the MC for which the cost is minimal.

ANNs are frequently used in reinforcement learning as part of the overall algorithm.

Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

## 3.4 Learning Rules

A Neural Network learns about its environment through an interactive process of adjustments applied to its synaptic weights and bias level.

Learing is a process by which the free parameters of a neural network get adapted through a process of simulation by the enviounment in which the network is embeded.The type of learning is detrmined by the manner in which the parameter changes takes place. The set of well defined rules for the solution of a learning problem is called learning algorithm. Various learning rules are descrbed below

### 3.4.1 Hebbian Learning Rule

Hebb's principle can be described as a method of determining how to alter the weights between model neurons. The weight between two neurons will increase if the two neurons activate simultaneously; it is reduced if they activate separately. Nodes which tend to be either both positive or both negative at the same time will have strong positive weights while those which tend to be opposite will have strong negative weights. It is sometimes stated more simply as "neurons that fire together, wire together."

If $x_j$ is the output of the presynaptic neuron, $x_j$ the output of the postsynaptic neuron, and $w_{ij}$ the strength of the connection between them, and $\gamma$. learning rate,.the one form of a learning rule would be:

$$\Delta W_{ij}(t)=\Upsilon * x_j * x_I \qquad (3.1)$$

A more general form of a hebbian learning rule would be:

$$\Delta W_{ij}(t) = F(xj, xi, \Upsilon, t, \theta) \qquad (3.2)$$

in which time and learning thresholds can be taken into account.

Hebbian learning has four features interesting to the cognitive scientist: first it is unsupervised; second it is a local learning rule, meaning that it can be applied to a network in parallel; third it is simple and therefore requires very little computation; fourth it is biologically plausible

### 3.4.2 Perceptron Learning Rule

This learning rule is an example of supervised training. is an input to the network and is the corresponding target output. As each input is applied to the network, the network

output is compared to the target. The learning rule then adjusts the weights and biases of the network in order to move the network output closer to the target.

The Perceptron Algorithm initialize the weights (either to zero or to a small random value) pick a learning rate ( this is a number between 0 and 1). Until stopping condition is satisfied (e.g. weights don't change):

For each training pattern (x, t):

Compute output activation y = f(w ,x)

If y = t, don't change weights

If y! = t, update the weights:

$$w\ (new) = w(old) + 2\ \mu\ t\ x \qquad (3.3)$$

or,

$$w(new) = w(old) + \mu(t - y\ )\ x,\ \text{for all t.} \qquad (3.4)$$

### 3.4.3 Delta Rule

Developed by Widrow and Hoff, the delta rule, also called the Least Mean Square (LMS) method, is one of the most commonly used learning rules. For a given input vector, the output vector is compared to the correct answer. If the difference is zero, no learning takes place; otherwise, the weights are adjusted to reduce this difference. The change in weight from $u_i$ to $u_j$ is given by: $dw_{ij} = r^* a_i * e_j$, where r is the learning rate, ai represents the activation of $u_i$ and $e_j$ is the difference between the expected output and the actual output of $u_j$. If the set of input patterns form a linearly independent set then arbitrary associations can be learned using the delta rule.

Construct a cost function E that measures how well the network has learned. For example

$$E = \frac{1}{2}\sum_{i=1}^{n} (t_i - y_i)^2 \qquad \text{(one output node)} \qquad (3.5)$$

where,

n = number of examples

$t_i$ = desired target value associated with the i-th example

$y_i$ = output of network when the i-th input pattern is presented to network

To train the network, we adjust the weights in the network so as to decrease the cost (this is where we require differentiability). This is called gradient descent.

Algorithm

- Initialize the weights with some small random value
- Until E is within desired tolerance, update the weights according to where E is evaluated at W(old), μ is the learning rate.: and the gradient is

$$W(new) = W(old) - \mu \frac{\partial E}{\partial W} \tag{3.6}$$

$$E = \frac{1}{2} \sum_{i=1}^{n} (t_i - y_i)^2 \tag{3.7}$$

$$\frac{\partial E}{\partial W} = \sum_{i=1}^{n} (t_i - y_i)\, x_i \tag{3.8}$$

### 3.4.4 Competitive Learning Rule

Competitive learning is a rule based on the idea that only one neuron from a given iteration in a given layer will fire at a time. Weights are adjusted such that only one neuron in a layer, for instance the output layer, fire. Competitive learning is useful for classification of input patterns into a discrete set of output classes. The "winner" of each iteration, element $i^*$ , is the element whose total weighted input is the largest. Using this notation, one example of a competitive learning rule can be defined mathematically as:

$$w_{ij}[n+1] = w_{ij}[n] + \Delta w_{ij}[n] \tag{3.9}$$

$$\Delta W_{ij}[n] = \{\eta(x_i - w_{ij}) \quad \text{if } i=j \tag{3.10}$$

$$0 \qquad \text{otherwise}\}$$

### 3.4.5 Boltzmann Learning Rule

Boltzmann learning is similar to an error-correction learning rule, in that an error signal is used to train the system in each iteration. However, instead of a direct difference between the result value and the desired value, we take the difference between the probability distributions of the system.

The neurons constitute a recurrent structure and they operate in a binary manner. The machine is characterized by an energy function E.

$$E = -\tfrac{1}{2}\Sigma_j\Sigma_k\, w_{kj}x_kx_j \;,\; j\neq k \qquad (3.11)$$

Machine operates by choosing a neuron at random then flipping the state of neuron k from state $x_k$ to state $-x_k$ at some temperature T with probability

$$P(x_k \rightarrow -x_k) = 1/(1+\exp(-\Delta E_k/T)) \qquad (3.12)$$

Clamped condition: the visible neurons are all clamped onto specific states determined by the environment

Free-running condition: all the neurons (=visible and hidden) are allowed to operate freely

The Boltzmann learning rule:

$$\Delta w_{kj} = \eta(\rho^{+}_{kj} - \rho^{-}_{kj}),\; j\neq k, \qquad (3.13)$$

note that both $\rho^{+}_{kj}$ and $\rho^{-}_{kj}$ range in value from –1 to +1.

## 3.5 Types of Artificial Neural Network

### 3.5.1 Perceptrons

The earliest kind of neural network is a single-layer perceptron network, which consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. In this way it can be considered the simplest kind of feed-forward network. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1).

A perceptron can be created using any values for the activated and deactivated states as long as the threshold value lies between the two. Most perceptrons have outputs of 1 or -1 with a threshold of 0 and there is some evidence that such networks can be trained more quickly than networks created from nodes with different activation and deactivation values.

Perceptrons can be trained by a simple learning algorithm that is usually called the delta rule. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent.

**Perceptron Architecture**

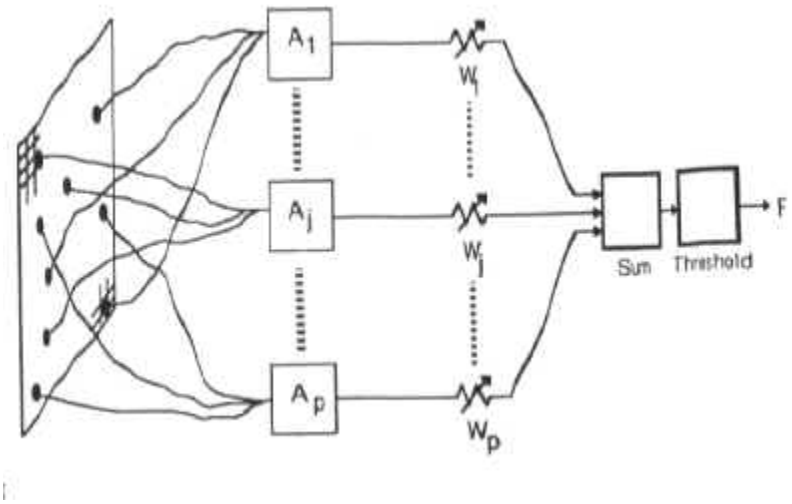


Figure. 3.3: Perceptron network

**Perceptron Training Algorithm**

The Perceptron Algorithm initialize the weights (either to zero or to a small random value) pick a learning rate (this is a number between 0 and 1). Until stopping condition is satisfied (e.g. weights don't change):

For each training pattern (x, t):

Compute output activation y = f(w x)

If y = t, don't change weights

If y ! = t, update the weights:

$$w(new) = w(old) + 2\ \mu\ t\ x$$

or

$$w(new) = w(old) + \mu(t - y)\ x, \text{ for all } t.$$

### 3.5.2 The Multilayer Perceptron Neural Network Model

The following diagram illustrates a perceptron network with three layers:

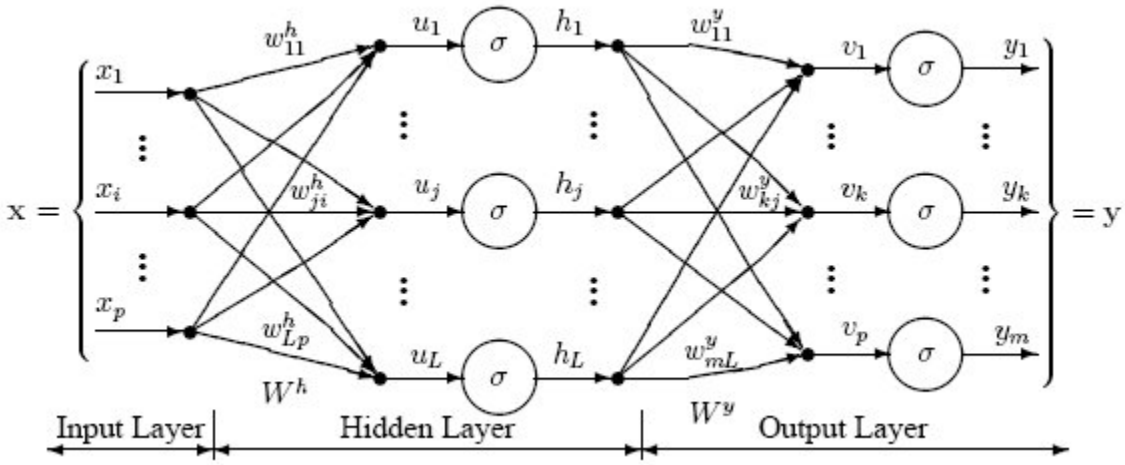


Figure. 3.4: Multilayer perceptron network

This network has an **input layer** (on the left) with three neurons, one **hidden layer** (in the middle) with three neurons and an **output layer** (on the right) with three neurons.

There is one neuron in the input layer for each predictor variable. In the case of categorical variables, N-1 neurons are used to represent the N categories of the variable.

**Input Layer** — A vector of predictor variable values ($x_1...x_p$) is presented to the input layer. The input layer (or processing before the input layer) standardizes these values by subtracting the median and dividing by the interquartile range and distributes the values to each of the neurons in the hidden layer. In addition to the predictor variables, there is a constant input of 1.0, called the bias that is fed to each of the hidden layers; the bias is multiplied by a weight and added to the sum going into the neuron.

**Hidden Layer** — Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by a weight ($w_{ji}$), and the resulting weighted values are added together producing a combined value $u_j$. The weighted sum ($u_j$) is fed into a transfer function, $\sigma$, which outputs a value $h_j$. The outputs from the hidden layer are distributed to the output layer.

**Output Layer** — Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by a weight ($w_{kj}$), and the resulting weighted values are added together producing a combined value $v_j$. The weighted sum ($v_j$) is fed into a transfer function, $\sigma$, which outputs a value $y_k$. The y values are the outputs of the network.

If a regression analysis is being performed with a continuous target variable, then there is a single neuron in the output layer, and it generates a single y value. For classification problems with categorical target variables, there are N neurons in the output layer producing N values, one for each of the N categories of the target variable.

**Multilayer Perceptron Architecture**

The network diagram shown above is a full connected, three layers, feed-forward, perceptron neural network. "Fully connected" means the output from each input and hidden neuron is distributed to all of the neurons in the following layer. "Feed forward" means that the values only move from input to hidden to output layers; no values are fed back to earlier layers (a Recurrent Network allows values to be fed backward).

All neural networks have an input layer and an output layer, but the number of hidden layers may vary. Here is a diagram of a perceptron network with two hidden layers and four total layers:

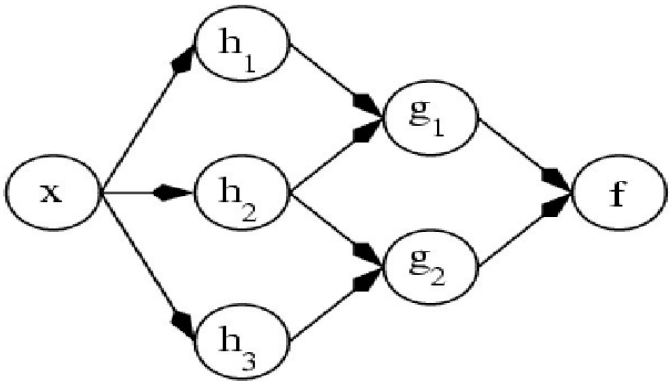

Figure 3.5: Multi-layer perceptron architecture

When there is more than one hidden layer, the output from one hidden layer is fed into the next hidden layer and separate weights are applied to the sum going into each layer.

**Training Multilayer Perceptron Networks**

The goal of the training process is to find the set of weight values that will cause the output from the neural network to match the actual target values as closely as possible. There are several issues involved in designing and training a multi-layer perceptron network:

- Selecting how many hidden layers to use in the network.
- Deciding how many neurons to use in each hidden layer.
- Finding a globally optimal solution that avoids local minima.
- Converging to an optimal solution in a reasonable period of time.
- Validating the neural network to test for over fitting.

### 3.5.3 ADALINE

ADALINE stands for Adaptive Linear Neuron. or later called Adaptive Linear Element. It's based on the McCulloch-Pitts model. It consists of a weight, a bias and a summation function. The Adaline has practical applications in the controls area. A single neuron with tap delayed inputs (the number of inputs is bounded by the lowest frequency present and the Nyquist rate) can be used to determine the higher order transfer function of a physical system via the bi-linear z-transform. This is done as the Adaline is, functionally, an

adaptive FIR filter. Like the single-layer perceptron, ADALINE has a counterpart in statistical modelling, in this case least squares regression.

An important generalization of the perceptron training algorithm was presented by Widrow and Hoff as the 'Least Mean Square' (LMS) learning procedure, also known as the delta rule. The main functional difference with the perceptron training rule is the way the output of the system is used in the learning rule. The perceptron learning rule uses the output of the threshold function (either -1 or +1) for learning. The delta-rule uses the net output without further mapping into output values -1 or +1. The learning rule was applied to the 'adaptive linear element,' also named Adaline2, developed by Widrow and Hoff (Widrow & Hoff, 1960).

**Adaline Architecture**

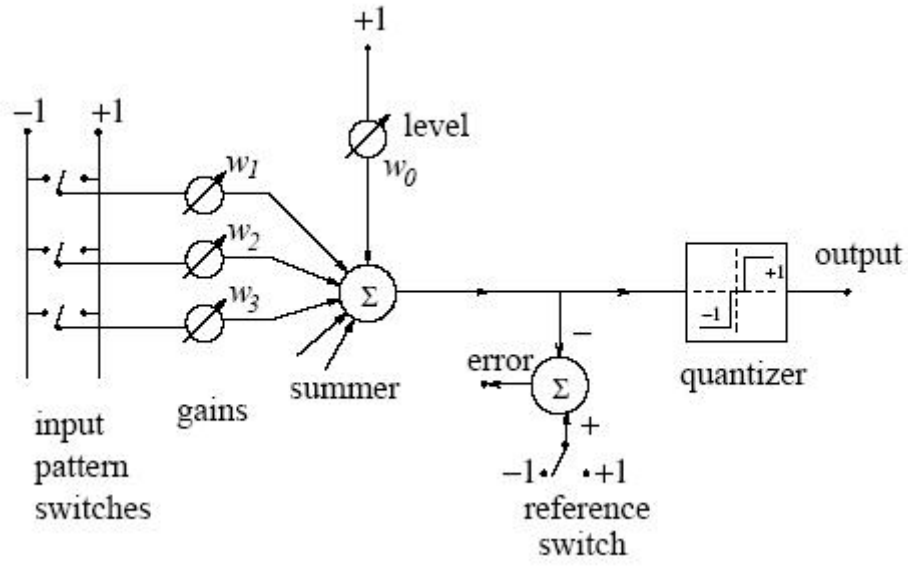


Figure 3.6: Architecture of adaline network

In a simple physical implementation this device consists of a set of controllable resistors connected to a circuit which can sum up currents caused by the input voltage signals. Usually the central block, the summer, is also followed by a quantiser which outputs either +1 of -1,depending on the polarity of the sum. Although the adaptive process is here exemplified in a case when there is only one output, it may be clear that a system with many parallel outputs is directly implementable by multiple units of the above kind.

If the input conductances are denoted by $w_i$, i = 0; 1; : : : ; n, and the input and output signals by $x_i$ and y, respectively, then the output of the central block is defined to be:

$$y = \sum_{i=1}^{n} w_i x_i + \theta, \tag{3.14}$$

where θ = w0. The purpose of this device is to yield a given value y = dp at its output when the set of values $x_i$ , i = 1,2..... , n, is applied at the inputs. The problem is to determine the coeficients $w_i$, i = 0, 1....., n, in such a way that the input-output response is correct for a large number of arbitrarily chosen signal sets. If an exact mapping is not possible, the average error must be minimised, for instance, in the sense of least squares. An adaptive operation means that there exists a mechanism by which the wi can be adjusted, usually iteratively, to attain the correct values.

Networks with linear activation functions: the delta rule

For a single layer network with an output unit with a linear activation function the output is simply given by:

$$y = \sum_{i=1}^{n} w_i x_i + \theta, \tag{3.15}$$

Such a simple network is able to represent a linear relationship between the value of the output unit and the value of the input units. By thresholding the output value, a classifier can be constructed (such as Widrow's Adaline), but here we focus on the linear relationship and use the network for a function approximation task. In high dimensional input spaces the network represents a (hyper)plane and it will be clear that also multiple output units may be defined.

**Training Algorithm**

Suppose we want to train the network such that a hyperplane is fitted as well as possible to a set of training samples consisting of input values $x^p$ and desired (or target) output values $d^p$. For every given input sample, the output of the network differs from the target value $d^p$ by $(d^p - y^p)$where $Y^p$ is the actual output for this pattern. The delta-rule now uses a cost- or error-function based on these differences to adjust the weights. The error

function, as indicated by the name least mean square, is the summed squared error. That is, the total error E is defined to be

$$E = \sum_p E^p = \tfrac{1}{2} \sum_p (d^p - y^p)^2, \tag{3.16}$$

where the index p ranges over the set of input patterns and $E^p$ represents the error on pattern p. The LMS procedure finds the values of all the weights that minimise the error function by a method called gradient descent. The idea is to make a change in the weight proportional to the negative of the derivative of the error as measured on the current pattern with respect to each weight:

$$\Delta_p w_j = -\gamma \frac{\partial E^p}{\partial w_j} \tag{3.17}$$

where γ is a constant of proportionality. The derivative is

$$\frac{\partial E^p}{\partial w_j} = \frac{\partial E^p}{\partial y^p} \frac{\partial y^p}{\partial w_j}. \tag{3.18}$$

Because of the linear units

$$\frac{\partial y^p}{\partial w_j} = x_j \tag{3.19}$$

$$\frac{\partial E^p}{\partial y^p} = -(d^p - y^p)$$

$$\Delta_p w_j = \gamma \delta^p x_j \tag{3.20}$$

where $\delta^p = d^p - y^p$ is the difference between the target output and the actual output for pattern p. The delta rule modifies weight appropriately for target and actual outputs of either polarity and for both continuous and binary input and output units. These characteristics have opened up a wealth of new applications.

### 3.5.4 Madaline

Madaline is the combination of adalines.It is also called multilayered adaline. If the adalines are combined, such that the output of some of them becomes input of others, then the net becomes multilayered. Madaline has two training algorithm, MRI and MRII.

In MRI, the weights of the hidden adaline units are only adjusted while the weights of the output units are fixed.In MRII, all weights present in the net are updated. Thus it allows traing of weights in all layers of net. The total error of any input pattern is given as the sum of squares of the errors at each output unit.Thie algorithm is different from MRI algorithm in the manner of the weight updation only.

**Madaline Architecture**

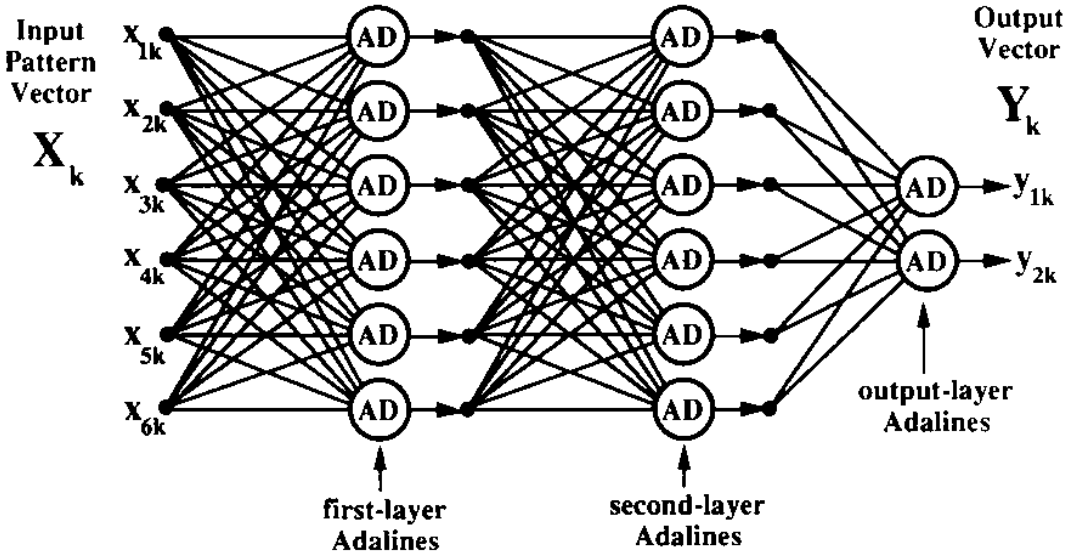


Figure 3.7: Architecture of madaline neural network

**MRI Algorithm**

Step1: Initialize weights and bias, set learning rate $\alpha$.

$V_1=V_2=0.5$ and $b_3=0.5$. Other weights may be small random values.

Step2: When stopping condition is false do steps 3-9.

Step3: For each bipolar training pair s:t, do steps 4-8.

Step 4: Set activations of input units:

$$X_i=si \text{ for } I = 1 \text{ to } n \qquad (3.21)$$

Step 5 : Calculate net input of hidden adaline units.

$$z_{-in1} = b_1 + x_1w_{11} + x_2w_{21} \qquad (3.22)$$

$$z_{-in2} = b_2 + x_1w_{12} + x_2w_{22} \qquad (3.23)$$

Step 6: Find output of hidden adaline unit using activation mentioned above.

$$Z_1 = f(z_{-in1}) \quad (3.24)$$

$$Z_2 = f(z_{-in2}) \quad (3.25)$$

Step 7: Calculate the net input to output

$$y_{-1n} = b_3 + z_1v_1 + z_2v_2 \quad (3.26)$$

Apply activation to get the output of net.

$$Y=f(y_{-in}) \quad (3.27)$$

Step 8: Find the error and do weight updation

If t=y, no weight updation

If t≠y, then,

If t=1, then update weigts on $z_j$ unit whose net input is closet to 0.

$$W_{ij}(new)= w_{ij}(old) + \alpha(1-Z_{inj})x_i \quad (3.28)$$

$$B_j(new) = b_j(old) + \alpha(1-Z_{inj}) \quad (3.29)$$

If t=-1, then update weigts on $z_k$ unit whose net input is closet to 0.

$$W_{ik}(new)= w_{ik}(old) + \alpha(1-Z_{ink})x_i \quad (3.30)$$

$$B_k(new) = b_k(old) + \alpha(1-Z_{ink}) \quad (3.31)$$

Step 9: Test for the stopping condition

**MRII Algorithm**

Step1: Initialize weights and bias, set learning rate α'.

All weights to a small random values.

Step2: When stopping condition is false do steps 3-11.

Step3: For each bipolar training pair s:t, do steps 4-10.

Step 4: Set activations of input units:

$$X_i=si \text{ for } I = 1 \text{ to } n$$

Step 5 : Calculate net input of hidden adaline units.

$$z_{-in1} = b_1 + x_1w_{11} + x_2w_{21}$$

$$z_{-in2} = b_2 + x_1w_{12} + x_2w_{22}$$

Step 6: Find output of hidden adaline unit using activation mentioned above.

$$Z_1 = f(z_{-in1})$$

$$Z_2 = f(z_{-in2})$$

Step 7: Calculate the net input to output

$$y_{-1n} = b_3 + z_1v_1 + z_2v_2$$

Apply activation to get the output of net.

$$Y=f(y_{-in})$$

Step 8: Find the error and do weight updation

If t=y, no weight updation

If t≠y, then,

Do steps 9=10 for each hidden unit whose net input is closet to 0, then for second closet etc.

Step 9 Change the units output

If t is +1, change to –1

If t is -1, change to +1

Step 10: Recompute the output of the net

If the error is reduced:

Adjust the weights on this unit.

Step 11: Test for the stopping condition

### 3.5.5 Associative

Pattern Association is the process of forming assocition between related patterns. The patterns may be of same type or of a different type.Associate neural Nets are single layer nets in which the weights are detrmined to store an asset of pattern associations.The associative nets mainly consists of two types of nets. If the input vector pair is same as the output vector pair, then it results in an auto associative net. If the input vector pair is different from that of the output vector pair, then it forms a hetero associative net.

**Auto**

Autoassociation is a means by which a neural network communicates that it does recognize the pattern that was presented to the network. A neural network that supports autoassociation will pass a pattern directly from its input neurons to the output neurons.

**Hetero**

Here weights are determined in such a way that the net can store a set of P pattern associations. Hetero associative networks are static networks. No non linear or delay or delay operations can be done.

## 3.6 Self Organizing Feature Map

### 3.6.1 Introduction

Self-organizing map (SOM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two dimensional), discretized representation of the input space of the training samples, called a map. The map seeks to preserve the topological properties of the input space. The model was first described as an artificial neural network by the Finnish professor Teuvo Kohonen, and is sometimes called a Kohonen map.Like most artificial neural networks, SOMs operate in two modes: training and mapping. Training builds the map using input examples. It is a competitive process, also called vector quantization. Mapping automatically classifies a new input vector.

The graphical representation of a self-organising map is usually a 2-dimenional, and sometimes 1-dimensional, gridded array in which there are a predefined number of elements, e.g. a 5 x 5 array will have 25 elements. The actual shape of the map can also vary according to the user's specifications. Each element can be referred to as an output node.

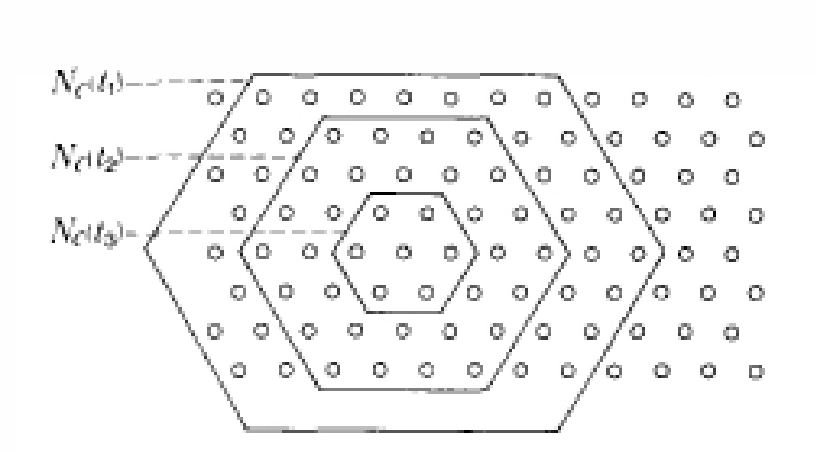


Figure 3.8: Neighborhood scheme for soms

### 3.6.2 Learning Algorithm

A SOM does not need a target output to be specified unlike many other types of network. Instead, where the node weights match the input vector, that area of the lattice is selectively optimized to more closely resemble the data for the class the input vector is a member of. From an initial distribution of random weights, and over many iterations, the SOM eventually settles into a map of stable zones. Each zone is effectively a feature classifier, so you can think of the graphical output as a type of feature map of the input space.

Training occurs in several steps and over many iterations:

- Each node's weights are initialized.
- A vector is chosen at random from the set of training data and presented to the lattice.
- Every node is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).
- The radius of the neighbourhood of the BMU is now calculated. This is a value that starts large, typically set to the 'radius' of the lattice, but diminishes each time-step. Any nodes found within this radius are deemed to be inside the BMU's neighbourhood.

- Each neighbouring node's (the nodes found in step 4) weights are adjusted to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered.
- Repeat step 2 for N iterations.

Now let's take a look at each step in detail.

**Step 1.Initializing The Weights**

Prior to training, each node's weights must be initialized. Typically these will be set to small standardized random values. The weights in the SOM from the accompanying code project are initialized so that $0 < w < 1$

**Step 2: Calculating the Best Matching Unit**

To determine the best matching unit, one method is to iterate through all the nodes and calculate the Euclidean distance between each node's weight vector and the current input vector. The node with a weight vector closest to the input vector is tagged as the BMU.

The Euclidean distance is given as:

$$\begin{aligned} J &= \min\{D_{ij}\}, \text{ for } j \in \Omega, \\ D_{ij} &= |P_i - W_j| = \sqrt{(x_i - w_{jx})^2 + (y_i - w_{jy})^2}, \end{aligned} \tag{3.32}$$

where $P_i=(x_i, y_i)$ is the current input vector and $W_j= (w_{jx},w_{jy})$ is the node's weight vector.

**Step 3: Determining the Best Matching Unit's Local Neighbourhood**

Each iteration, after the BMU has been determined, the next step is to calculate which of the other nodes are within the BMU's neighbourhood. All these nodes will have their weight vectors altered in the next step. So, first you calculate what the radius of the neighbourhood should be and then it's a simple application of Pythagoras to determine if each node is within the radial distance or not.

A unique feature of the Kohonen learning algorithm is that the area of the neighbourhood shrinks over time. This is accomplished by making the radius of the neighbourhood shrink over, using the exponential decay function:

$$\sigma(t) = \sigma_0 \exp(-t/\lambda) \qquad t=1,2,3\ldots\ldots \qquad (3.33)$$

where the Greek letter sigma, $\sigma_0$ , denotes the width of the lattice at time $t_0$ and the Greek letter lambda,$\lambda$, denotes a time constant. t is the current time-step (iteration of the loop )

The value of $\lambda$ is dependent on $\sigma$ and the number of iterations chosen for the algorithm to run. 'n'is the number of iterations the learning algorithm will perform. This value is set by the user.

If a node is found to be within the neighbourhood then its weight vector is adjusted as follows

**Adjusting the Weights**

Every node within the BMU's neighbourhood (including the BMU) has its weight vector adjusted according to the following equation:

$$L(t) = L_0 \exp(-t/\lambda) \qquad t = 1,2,3\ldots\ldots \qquad (3.34)$$

Where t represents the time-step and L is a small variable called the learning rate, which decreases with time. Basically, what this equation is saying, is that the new adjusted weight for the node is equal to the old weight (W), plus a fraction of the difference (L) between the old weight and the input vector (V).

The decay of the learning rate is calculated each iteration using the following equation:

$$W(t+1) = W(t) + L(t)(V(t)-W(t)) \qquad (3.35)$$

The learning rate at the start of training,is set as 0.1. It then gradually decays over time so that during the last few iterations it is close to zero.

Not only does the learning rate have to decay over time, but also, the effect of learning should be proportional to the distance a node is from the BMU. Indeed, at the edges of the BMUs neighbourhood, the learning process should have barely any effect. Ideally, the

amount of learning should fade over distance similar to the Gaussian decay. To achieve this, all it takes is a slight adjustment to Equation 3.34.

$$W(t+1) = W(t) + \theta(t)L(t)(V(t)-W(t)) \qquad (3.36)$$

The Greek capital letter theta, $\theta$ ,is used to represent the amount of influence a node's distance from the BMU has on its learning. $\theta$ (t) is given by Equation 3.37.

$$\theta(t) = \exp(- dist^2 /2\sigma^2(t)) \qquad t= 1,2 ,3 \ldots\ldots \qquad (3.37)$$

Where dist is the distance a node is from the BMU and $\sigma$, is the width of the neighbourhood function as calculated by Equation 2. Additionally, please note that $\theta$ also decays over time.

### 3.6.3 The SOM Algorithm - A Summary of Steps

1. Initialise the topology of the map. This may mean initialising the weight vectors to randomised values.
2. Decide on appropriate g and r values (and neighbourhood criterion)
3. For each input node:
   Find the shortest distance to any output node.
4. Modify the winner node's weight according to the current state of g.
5. Modify the neighbouring node's weights according to the current state of r.
6. Go to the next unvisited input node. If there are no unvisited input nodes left then go back to the very first one and go to Step 4.
7. When a previously visited node has been reached, incrementally decrease g and r and repeat Step 3.
8. Keep doing Steps 3 and 4 for a sufficient number of iterations.

## 3.7 Application of SOM in Robotics

In the field of robotics, the SOM acts attention as an efficient tool to realize robot intelligence, e.g., Ritter and his group has been investigating the research on the direct experimental approaches to elucidate the architecture of higher brains associated with the possibilities and limits of artificial control architectures for robot systems. And the techniques of SOM are applied into problems such as posture analysis self-location, collision avoidance, path planning and so on, and the results show the robustness and adaptability.

# Chapter 4

## Problem Formulation

The main objective of this work is to find an optimized path for each robot to reach to the desired location (target location). Here, optimized is a path, which covers maximum distance in minimum time, hence, which follows shortest distance method. Once an optimized path has been selected, the task will be assigned to each robot. So, in this work, path planning and task assignment both are done simultaneously. There is no need for each robot to reach to the target and assign the task. Hence the objective is to find and apply an efficient algorithm, to assign the robot, the tasks and target location and to control motion of multi mobile robot. Also, through these algorithms, the effect of change (increase or decrease) of learning rate$\chi$ is also studied. So, consider two types of case study:

In case study I, first path planning is done and then tasks are assigned to each robot. Under this study, the change in environment and requirement has been considered. Such as, if any of the robots breaks down, or if any of the target location needs more than one robot, or any case where targets are also movable. The system should work in all possible circumstances.

In case study II, the effect of learning rate on the movement of robot has been studied. The change i.e. increase or decrease in rate will cause change in speed of robot, and it is to see how? The parameter will change and will try to find the range within which robot can work efficiently.

In the end, a system will be found which will be efficient enough to give path planning, task assignment, and control over robot at best learning and gain rate.

## 4.1 Case Study I

In this an optimized path has been assigned to each robot. Under this case study different types of cases that are studied are: -

**Case 1:** It's a case when in a workspace, the number of robots and the number of targets are equal. Let here R=5 and T=5, as shown in figure 4.1 below:

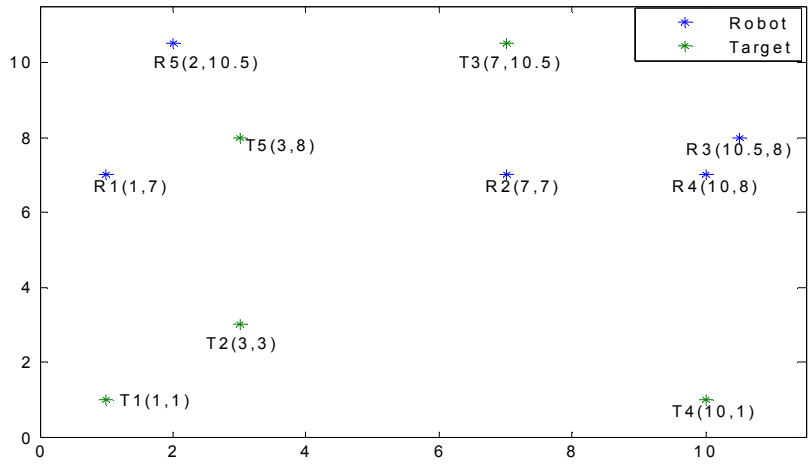


Figure 4.1: Case when R= T

**Case 2:** When in a common workspace number of robots are more than number of targets. And different target locations require, different number of targets, can be one or more. Let here R= 9 and T= 5. Case has been shown below in the figure 4.2.

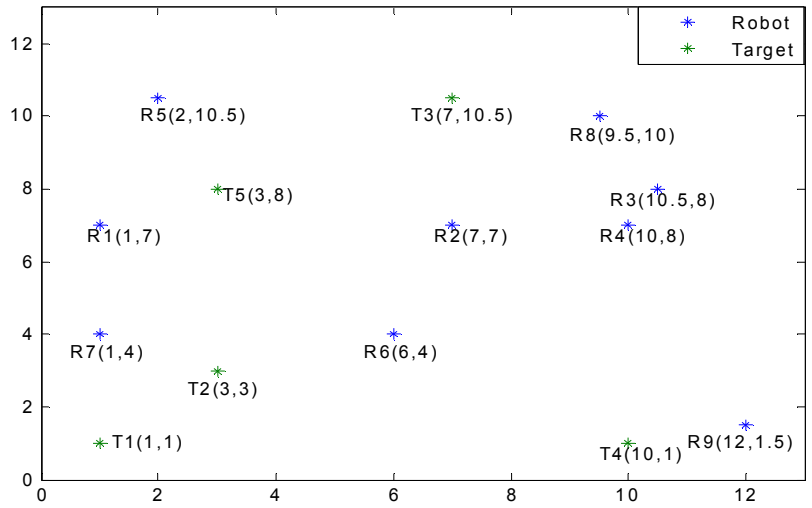


Figure 4.2: Case when R>T

**Case 3:** If any robot breaks down or fails to do its operation. It will be a case where targets will be more than robots. So, same robot will reach to different target location, after finishing its previous task. Figure 4.3 shows, R=3 and T=5

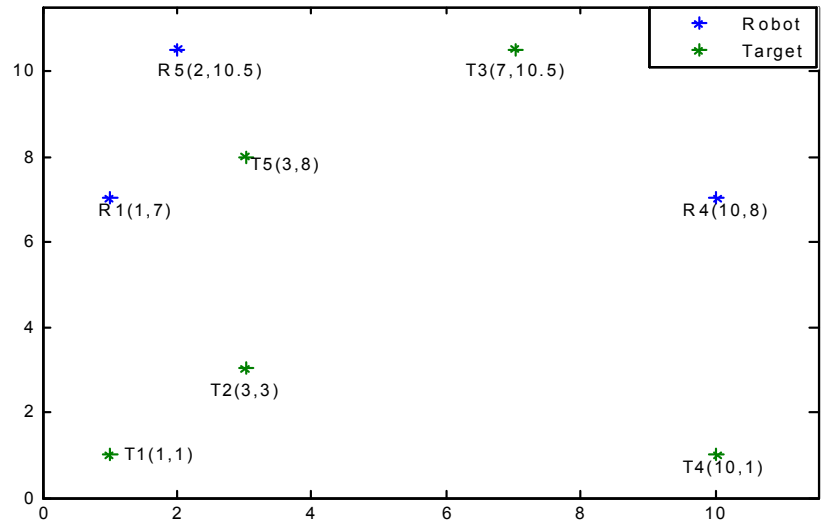


Figure 4.3: Case when R<T

**Case 4:** When target locations will be movable at constant speed, but will be slower than robot, shown in Figure 4.4 below:

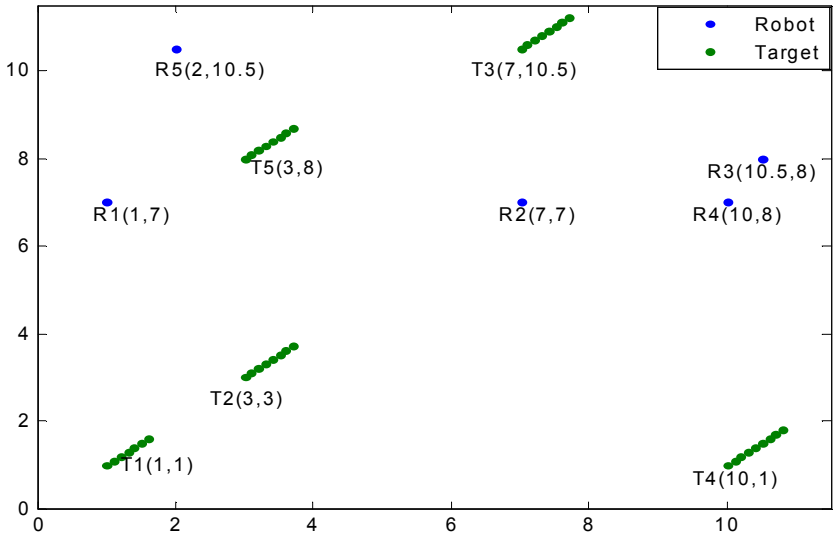


Figure 4.4: Case when T(x1,y1)→ T(x2,y2)→ T(x3,y3)

To solve all above cases, the neural network approach for self-organizing map has been used. Each target location has been considered as nodes of input layer and each robot is considered as output node. The location or coordinate axes of these nodes are called as weight. During the training of this neural network, these weights are updated and hence coordinate axes of robot will update.

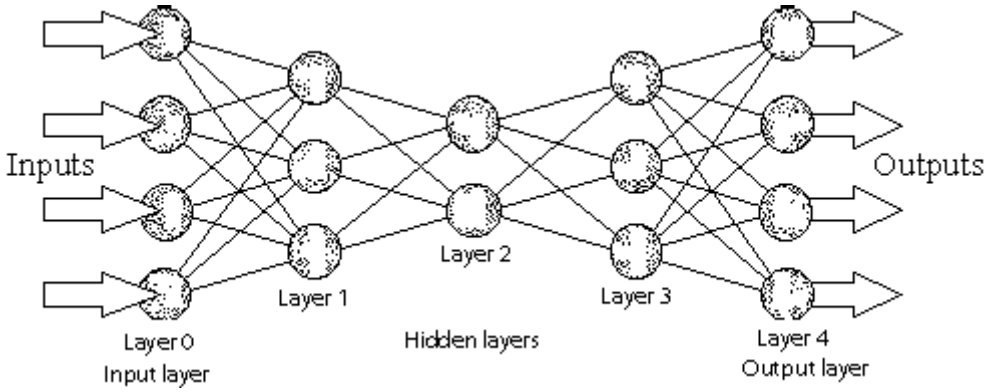


Figure 4.5: Neural network architecture

According to the weight updations the robot will move and hence will follow the path. This procedure is called as Path Planning. Before defining its algorithm, some assumptions are to be done. The algorithm that we will define will be for a general case where any can be the number of Robots and also, can be any number of Targets.
Let us assume there are 'l' number of targets and 'm' number of robots in a workspace. The input layer will have only one node( only one target input will be given at one time) and, all robots will be in the output node.

## Algorithm

To assign the task and decide the path we will follow the following algorithm:

Step 1: - Select the target location where robot is needed. Give co-ordinates of the target as input to the input node of shown Neural Network.

Ti (x, y) where, i = 1,2,3,……l, no.of Targets.

Step 2: - Assign the value to the each output node, write co-ordinates of each robot.

Rj (x, y) where, j =1,2,3……..….m, no. of Robots.

Step 3: - Calculate the distance of each robot from the input target location using the relation.

$$Dij = \sqrt{(Tix-Rjx)2 +(Tiy-Rjy)2} \qquad (4.1)$$

Step 4: - To decide the winner node find the minimum distance among all the distances calculated in Step 3

$$J= \min ( Dij) \qquad (4.2)$$

Step 5: -The neighboring node will be the node whose distance will be next to the winner node's distance.

Step 6: - Now, calculate the neighborhood function using the relation

$$N(Dj, A) = \{ \ e(-Dj2/A2), \quad \text{if } Dj < al \qquad (4.3)$$

$$0, \quad \text{otherwise}\}$$

Where Dj=0, for winner node, Dj= 1, for the next nearest node and so on….

Also, $A=(1-\lambda)^r A$, Where A is intiated as A=1

$\lambda=\lambda(0.998)r$, Where $\lambda$ is intiated as 0.05

a is a small constant

Where A is gain prameter and $\lambda$ is gain parameter change rate and also r is the number of iterations.

Step 7:- Updation of weights. The following formula will be used

$$Rj = \{Ti, \quad \text{if } Dij < \mu \min Dij \qquad (4.4)$$

$$Rj + \chi \, N(Dj, A) \, (Ti-Rj), \quad \text{otherwise } \}$$

Where $\chi$ is learning rate and is initialized to value 0.5, $\mu$ is a small constant (i.e. less than 0.5)

### 4.2 Case study II

The values of learning rate $\chi$, have been varied from 0.05 to1 at a constant interval of 0.1. Hence different path planning will be obtained for 10 cases varying $\chi$ as 0.05, 0.06, 0.07……..so on till 1.And other parameters will remain constant. Here $\gamma$ =0.03, A=1 and $\lambda$=0.2. It will be applied to only one input. As, the result will be same for all cases.

# Chapter 5

## Simulation and Testing

Till now many different cases have been seen, and the possible algorithm to find the best fitted solution.

Now, each case and each input has been dealt separately.

### 5.1 Case I

It's a case in which number of inputs i.e. target locations are exactly equal to the outputs i.e. Robots. So, according to the input target location a particular robot will be selected on the basis of "minimum distance travel method". Once a robot has been selected for a particular target location and a task have been assigned, it could not be reselected. So, it will be kept out of competition.

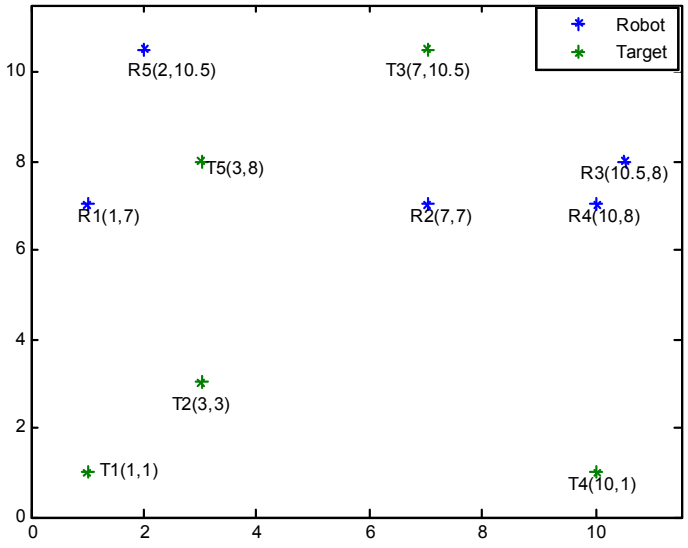


Figure 5.1:Workspace for case 1, when R=T

## Input 1

Let the first input is target location $T_1$ (1,1). Now its distance will be found from each robot and on the basis of minimum distance travel method, a particular robot will be selected. For different robots there will be different distance measurement. As shown in table 5.1 below:

Table 5.1:Disatnce measurement for input 1

| Robot | Distance | |
|---|---|---|
| $R_1$ (1,7) | D11 | 6 |
| $R_2$ (7,7) | D12 | 8.48 |
| $R_3$ (10.5,8) | D13 | 11.8 |
| $R_4$ (10,7) | D14 | 10.81 |
| $R_5$ (2,10.5) | D15 | 9.55 |

Here, minimum distance found is 6.And the distance next to it is 8.485. So, for target $T_1$, the robot $R_1$ is the winner and robot $R_2$ is the neighbour. Now the approximate path which these robots could follow will be found. Hence the path to be followed will be:

Table 5.2: Path planning for input 1

| | |
|---|---|
| $R_1$ | (1,7)(1,4)(1,2.5)(1,1.75)(1,1.375)(1,1.1875)(1,1.09375)(1,1.04687)(1,1.02343)(1,1.0117) (1,1.00585)(1.1.0029)(1,1.0004)(1, 1.0002)(1,1.0001)(1,1). |
| $R_2$ | (7,7)(4,4)(2.5,2.5)(1.75,1.75)(1.375,1.375)(1.1875,1.1875)(1.0938,1.0938)(1.0469,1.0469) (1.0234,1.0234)(1.0117,1.0117)(1.0059,1.0059)(1.0029,1.0029)(1.0015,1.0015)(1.0007,1.0007) (1.0004,1.0004)(1.0002,1.0002)(1.0001,1.0001)(1,1). |

Table 5.2, showed that the number of iterations used by $R_1$ are (16) less than that of neighbour $R_2$(18).So, neighbour $R_2$ will be neglected.

## Input 2

Now putting the next input let, it be target location $T_2(3,3)$.Again its distance from all the robots will be found exept from robot $R_1$ as it was selected in the previous case.For Rest of the robots the distance found is tabulated bellow in Table 5.3,

Table 5.3: Distance measurement for input 2

| Robot | Distance | |
|---|---|---|
| R2(7,7) | D22 | 5.6 |
| R3(10.5,8) | D23 | 9 |
| R4(10,7) | D24 | 8 |
| R5(2,10.5) | D25 | 7.5 |

Again finding the minimum distance among all the distances . Here minimum distance found is 5.6 and the distance next to it is 7.5.So, for input 2, the robot $R_2$ is the winner and $R_5$ is its neighbour. Now the approximate path for these two robots will be found, and it will be as follows

Table 5.4 : Path planning for input 2

| | |
|---|---|
| $R_2$ | (7,7)(5,5)(4,4)(3.5,3.5)(3.25,3.25)(3.125,3.125)(3.0625,3.0625)(3.0313,3.0313)(3.0156,3.0156) (3.0078,3.0078)(3.0039,3.0039)(3.0020,3.0020)(3.0010,3.0010)(3.0005,3.0005)(3.0002,3.0002) (3.0001,3.0001)(3,3) |
| $R_5$ | (2,10.5)( 2.5,6.75)(2.75,4.875)(2.875,3.9375)(2.9375,3.4688)(2.9688,3.2344)(2.9844,3.1172) (2.9922,3.0586)(2.9961,3.0293)(2.9980,3.0146)(2.9990,3.0073)(2.9995,3.0037)(2.9998,3.0018) (2.9999,3.0009)(2.9999,3.0005)(3,3.0002)(3,3.0001)(3,3.0001)(3,3) |

Above table 5.4 says that number of iterations in the case of R2(16) are less than those in the case of R4. So, only robot R2 will to the target location T2.

## Input 3

Next input target location is T3, it is located at (7,10.5). Now Robots R1 and R2 both are kept out of the competition. So, the distance of the target will be measured from rest of the three robots. Different distance measurements for different robots are given in table 5.5.

Table 5.5: Distance measurement for input 3

| Robot | Distance | |
|---|---|---|
| R3(10.5,8) | D33 | 4.3 |
| R4(10,7) | D34 | 4.6 |
| R5(2,10.5) | D35 | 5 |

The minimum distance found among all the distances is 4.3 and next to it is 4.6.
So, robot R3 is the winner and the robot R4 is its neighbour. Next step is to calculate the path that will be followed by these two robots

Table 5.6: Path planning for Input 3

| | |
|---|---|
| $R_3$ | (10.5,8)(8.75.9.25)(7.875,9.875)(7.4375,10.1875)(7.2188,10.3438)(7.1094,10.4219)(7.0547,10.4609) (7.0273,10.4805)(7.0137,10.4902)(7.0068,10.4951)(7.0034,10.4976)(7.0017,10.4988)(7.0009, 10.4994)(7.0004,10.4997)(7.0002,10.4998)(7.0001,10.4999)(7,10.5). |
| $R_4$ | (10,7)( 8.5,8.75)(7.75,9.625)(7.375,10.0625)(7.1875,10.2813)(7.0938,10.3906)(7.0469,10.4453) (7.0234,10.4727)(7.0117,10.4863)(7.0059,10.4932)(7.0029,10.4966)(7.0015,10.4983)(7.0007,10.4991) (7.0004,10.4996)(7.0002,10.4998)(7.0001,10.4999)(7,10.4999)(7,10.5) |

The number of iterations followed by R3(16) are less than that of R5(18). So, Robot R3 has been selected for the input target. Now, R3 will also be kept out of the competion.

## Input 4

Out of the rest of the two target location, now the input is target T4 loacted at (10,1). Only two robots are left and we will find minimum distance of these two robots from target.

Table 5.7: Distance measurement for input 4

| Robot | Distance | |
|---|---|---|
| R4(10,7) | D44 | 6 |
| R5(2,10.5) | D45 | 12.3 |

Table 5.7 shows that R4 is at less distance from target becomes winner and R5 will be neighbour. Now, let us calculate their feasible path to the target. The path to be followed is shown in Table 5.8.

Table 5.8:Path planning for input 4

| | |
|---|---|
| $R_4$ | (10,7)(10,4)(10,2.5)(10,1.75)(10,1.375)(10,1.1875)(10,1.0938)(10,1.0469)(10,1.0234)(10,1.0117) (10,1.0059)(10,1.0029)(10,1.0015)(10,1.0007)(10,1.0004)(10,1.0002)(10,1.0001)(10,1). |
| $R_5$ | (2,10.5)(6,5.75)(8,3.375)(9,2.1875)(9.5,1.5938)(9.75,1.2969)(9.875,1.1484)(9.9375,1.0742) (9.9688,1.0371)(9.9844,1.0186)(9.9922,1.0093)(9.9961,1.0046)(9.998,1.0023)(9.999,1.0012) (9.9995,1.0006)(9.9998,1.0003)(9.9999,1.0001)(9.9999,1.0001)(10,1) |

R4 (17)will reach to target in less time and less steps so it is selected for this input target T4.

## Input 5

This is the final and last input. As here there were equal number of targets and robots so, this last target location T5 (3,8) has only one robot R5 (2,10.5) left and hence selected. The path to be followed by robot R5 is

Table 5.9: Path planning for input 5

| | |
|---|---|
| $R_5$ | (2,10.5)(2.5,9.25)(2.75,8.625)(2.8750,8.3125)(2.9375,8.1563)(2.9688,8.0781)(2.9844,8.0391)<br>(2.9922,8.0195)(2.9961,8.0098)(2.9980,8.0049)<br>(2.9990,8.0024)(2.9995,8.0012)(2.9998,8.0006)(2.9999,8.0003)(2.9999,8.0002)(3,8.0001)(3,8). |

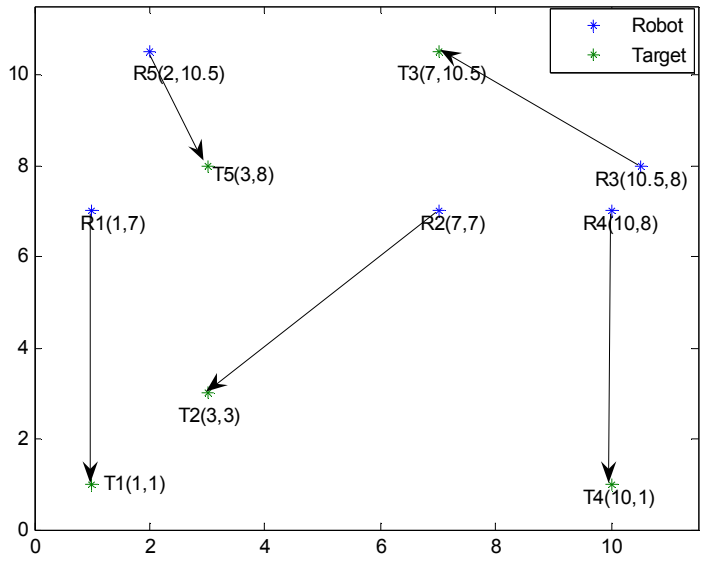


Figure 5.2: Path planning of case 1

The all above calculations that we have done for Case 1 (when R=T) is tabulated in Table 5.10.

Table 5.10: Path Planning and task assignment of all targets for case 1

| Input(Given Target loaction) | Winner node | Neighbour node | No. of iterations required,resp. | Output(Selected Robot) |
|---|---|---|---|---|
| T1(1,1) | $R_1$ (1,7) | $R_2$ (7,7) | 17 | $R_1$ (1,7) |
| T2(3,3) | $R_2$ (7,7) | $R_4$ (10,7) | 16 | $R_2$ (7,7) |
| T3(7,10.5) | $R_3$ (10.5,8) | $R_5$ (2,10.5) | 16 | $R_3$ (10.5,8) |
| T4(10,1) | $R_4$ (10,7) | $R_4$ (10,7) | 18 | $R_4$ (10,7) |
| T5(3,8) | $R_5$ (2,10.5) | | 17 | |

## 5.2 Case II

The other case to be consider is if number of target locations and more than number of robots available, or we can describe it in different ways such that if some of the robots stop working and they ar e required at various target locations.

How we will handle this particular case??

Here once a robot will move to input target location . For next input it will serve its operation from the new location(the location of the previous target location). Hence even after working for a particular particular target it will remain active. Let us now deal with this particular case for different target input locations.

Considering previous case again but this time out of Five robots only three are are working and two are failed( unabe to perform the task). Let Robot R1,R4, and R5 are in the race and robot R2 and R3 are failed. So for all five target locations there are only 3 robots available.

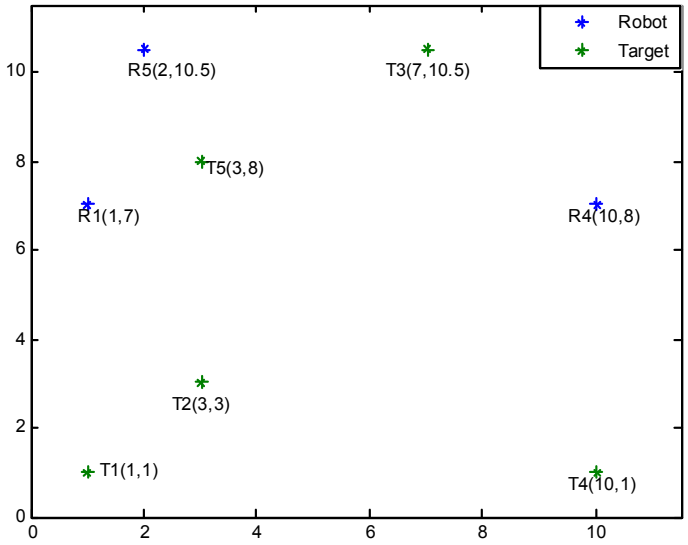


Figure 5.3: Workspace for case 2 when R<T

## Input 1

As similar to the previous case, the first input target location is T1(1,1). To select a particular robot for this input target, we will calculate the distance of all robots from the target. And the robot with minimum distance will be the winner and the next minimum distant robot will be neighbor. These distances are tabulated below

Table 5.11: Distance measurement for input 1

| Robot | Distance | |
|---|---|---|
| R1(1,7) | D11 | 6 |
| R4(10,7) | D14 | 10.80 |
| R5(2,10.5) | D15 | 9.5 |

Among all the distances R1 has the minimum distance and R4 has greater than it. Hence R1 is winner and R5 is the neighbour. Applying the neighbourhood function to the to the two robots we will find their approximate path planning as given below

Table 5.12:Path planning for input 1

| | |
|---|---|
| R1 | (1,7)(1,4)(1,2.5)(1,1.75)(1,1.375)(1,1.1875)(1,1.09375)(1,1.04687)(1,1.02343)(1,1.0117) (1,1.00585)(1.1.0029)(1,1.0004)(1, 1.0002)(1,1.0001)(1,1). |
| R5 | (2,10.5)(1.6728,7.3912)(1.4526,5.2998)(1.3045,3.8927)(1.2049,2.9461)(1.1378,2.3093) (1.0927,1.8808)(1.0624,1.5926)(1.0420,1.3987)(1.0282,1.2682)(1.0190,1.1804)(1.0128, 1.1214)(1.0086,1.0817)(1.0058,1.0549)(1.0039,1.0370)(1.0026,1.0249)(1.0018,1.0167) (1.0012,1.0113)(1.0008,1.0076)(1.0005,1.0051)(1.0004,1.0034)(1.0002,1.0023)(1.0002, 1.0016)(1.0001,1.0010)(1.0001,1.0007)(1,1.0005)(1,1.002)(1,1.001)(1,1) |

Again R1 will cover the whole distance in fewer steps. So, it will perform the desire task at desire location. After reaching to the target the new location of robot R1 will be (3,3).

## Input 2

Now, next input will be given i.e. next target location is specified. The input is target 2 located at (3,3). Here in this case all Robots can be reused. So, we will find the distance of T2 from all robots, where robot R1 has new location (1,1), rest all robots are at same location. Now, distance of target T2 from all robots is tabulated below

Table 5.13: Distance measurement for input 2

| Robot | Distance | |
|---|---|---|
| R1(1,1) | D21 | 2.82 |
| R4(10,1) | D24 | 8.04 |
| R5(2,10.5) | D25 | 7.56 |

Now we will select the winner on the basis of minimum distance method. Here for input T2, R1 is at minimum distance and R5 is next to it. Again by applying the neighbourhood function to the two robots the path will be planned.

Table 5.14:Path planning for input 2

| | |
|---|---|
| R1 | (1,1)(2,2)(2.5,2.5)(2.75,2.75)(2.875,2.875)(2.9375,2.9375)(2.9688,2.9688)(2.9844,2.9844) 2.9922,2.9922)(2.9961,2.9961)(2.9980,2.9980)(2.9990,2.9990)(2.9995,2.9995)(2.9998,2.9998)(3,3) |
| R5 | (2,10.5)(2.3272,8.0457)(2.5474,6.3945)(2.6955,5.2837)(2.7951,4.5364)(2.8622,4.0336)(2.9073, 3.6954)(2.9376,3.4678)(2.958,3.3147)(2.9718,3.2117)(2.981,3.1425)(2.9872,3.0958)(2.9914,3.0645) (2.9942,3.0434)(2.9961,3.0292)(2.9974,3.0196)(2.9982,3.0132)(2.9988,3.0089)(2.9992,3.0060) (2.9995,3.0040)(2.9996,3.0027)(2.9998,3.0018)(2.9998,3.0012)(2.9999,3.0008)(2.9999,3.0006) (3,3.0004)(3,3.002)(3,3.001)(3,3) |

We will follow only one path i.e. the path will be planned for Robot R1. After reaching to the target the new location of Robot R1 will be (3,3).

## Input 3

Next input target location is T3, and it is located at (7,10.5). Now we will measure its distance from all the robots.

Table 5.15: Distance measurement for input 3

| Robot | Distance | |
|---|---|---|
| R1(3,3) | D31 | 8.47 |
| R4(10,7) | D34 | 4.58 |
| R5(2,10.5) | D35 | 5 |

The minimum distance among all the distances is of robot R4 and next to it is robot R5. Hence, winner is robot R4 and neighbour is robot R5. Seeing the approximate path that these robot should follow

Table 5.16: Path planning for input 3

| | |
|---|---|
| R4 | (10,7)(8.5,8.75)(7.75,9.625)(7.375,10.0625)(7.1875,10.2813)(7.0938,10.3906)(7.0469,10.4453) (7.0234,10.4727)(7.0117,10.4863)(7.0059,10.4932)(7.0029,10.4966)(7.0015,10.4983) (7.0007,10.4991)(7.0004,10.4996)(7.0002,10.4998)(7.0001,10.4999)((7.0000,10.4999)(7,10.5) |
| R5 | (2,10.5)(3.6362,10.5)(4.737,10.5)(5.4775,10.5)(5.9757,10.5)(6.3109,10.5)(6.5364,10.5)(6.6881,10.5) |

| | |
|---|---|
| | (6.7902,10.5)(6.8588,10.5)(6.905,10.5)(6.9361,10.5)(6.957,10.5)(6.9711,10.5)(6.9805,10.5)(6.9869, 10.5)(6.9912,10.5)(6.9941,10.5)(6.996,10.5)(6.9973,10.5)(6.9982,10.5)(6.9988,10.5)(6.9992,10.5) (6.9995,10.5)(6.9996,10.5)(6.9998,10.5)(6.9999,10.5)(7,10.5) |

R4 will reach upto the target in less number of steps hence task will be assigned to it.

## Input 4

As similar to previous case, next input is trget location T4 at (10,1). Again for this input all robots are active. New location of robot R4 is (7,10.5). Finding again the distance of all robots from T4(10,1).Table 5.17 shows all distances

Table 5.17: Distance measurement for input 4

| Robot | Distance | |
|---|---|---|
| R1(3,3) | D31 | 7.28 |
| R4(7,10.5) | D34 | 9.962 |
| R5(2,10.5) | D35 | 12.41 |

The robot R1 is at minimum distance from the input target T4 and R4 is next to it. Hence, according to above defined rules, R1 is the winner and R4 is the runner up hence, neighbor. Now, finding the approximate path that these robots should follow.

Table 5.18:Path planning for input 4

| | |
|---|---|
| R1 | (3,3)(6.5,2)(8.25,1.5)(9.125,1.25)(9.5625,1.1250)(9.7813,1.0625)(9.8906,1.0313) (9.9453,1.0156)(9.9727,1.0078)(9.9863,1.0039)(9.9932,1.002)(9.9966,1.001)(9.9983,1.0005) (9.9991,1.0002)(9.9996,1.0001)(9.9998,1.0001)(9.9999,1)(9.9999,1)(10,1) |
| R4 | (7,10.5)(7.9817,7.3912)(8.6422,5.2998)(9.0865,3.8927)(9.3854,2.9461)(9.5866,2.3093)(9.7218, 1.8808)(9.8129,1.5926)(9.8741,1.3987)(9.9153,1.2682)(9.9430,1.1804)(9.9617,1.1214)(9.9742, 1.0817)(9.9826,1.0549)(9.9883,1.0370)(9.9921,1.0249)(9.9947,1.0167)(9.9964,1.0113)(9.9976, 1.0076)(9.9984,1.0051)(9.9989,1.0034)(9.9993,1.0023)(9.9995,1.0016)(9.9997,1.0010)(9.9998, 1.0007)(9.9999,1.0005)(9.9999,1.0003)(9.9999,1.0002)(10,1.0001)(10,1.0001)(10,1) |

R1 will perform its task at target T4, as it will reach at the location in time

## Input 5

The last input will be target location T5 and for this also all inputs are active. The active robots will be R1 located at (10,1), R4 located at (7,10.5) and R5 located at (2,10.5). Finding the distance of all these robots from th in put target located at(3,8).

Table 5.19: Distance measurement for input 5

| Robot | Distance | |
|---|---|---|
| R1(10,1) | D51 | 9.86 |
| R4(7,10.5) | D54 | 4.71 |
| R5(2,10.5) | D55 | 2.7 |

The robot R1 is found to be at minimum distance. Hence called as winner and the robot R4 is neighbor. The path planning of these two robot is shown in table 5.20

Table 5.20: Path planning for input 5

| | |
|---|---|
| R5 | (2,10.5)(2.5,9.25)(2.75,8.625)(2.8750,8.3125)(2.9375,8.1563)(2.9688,8.0781)(2.9844,8.0391) (2.9922,8.0195)(2.9961,8.0098)(2.9980,8.0049) (2.9990,8.0024)(2.9995,8.0012)(2.9998,8.0006)(2.9999,8.0003)(2.9999,8.0002)(3,8.0001)(3,8). |
| R4 | (7,10.5)(5.691,9.6819)(4.8104,9.1315)(4.218,8.7612)(3.8194,8.5121)(3.5513,8.3445)(3.3709, 8.2318)(3.2495,8.1559)(3.1679,8.1049)(3.1129,8.0706)(3.0760,8.0475)(3.0511,8.0319)(3.0344, 8.0215)(3.0231,8.0145)(3.0156,8.0097)(3.0105,8.0065)(3.007,8.0044)(3.0047,8.003)(3.0032,8.0020) (3.0021,8.0013)(3.0014,8.0009)(3.001,8.0006)(3.0007,8.0004)(3.0004,8.0003)(3.0003,8.0002) (3.0002,8.0001)(3.0001,8.0001)(3.0001,8.0001)(3,8) |

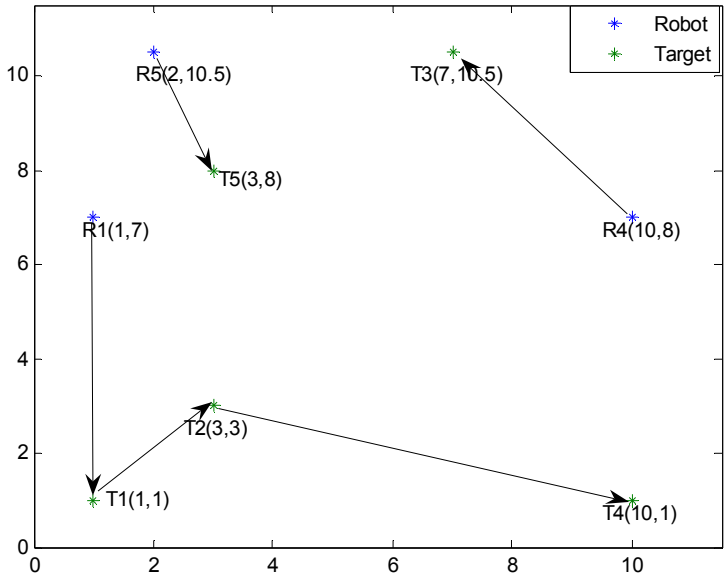


Figure 5.4: Path planning of case 2

The all above calculations done for Case 2 (when R<T) is tabulated in Table 5.20.

Table 5.21: Path Planning and task assignment of all targets for case 2

| Input(Given Target loaction) | Winner node | Neighbour node | No. of iterations required,resp. | Output(Selected Robot) |
|---|---|---|---|---|
| T1(1,1) | $R_1$ (1,7) | $R_4$ (10,7) | 17 | $R_1$ (1,7) |
| T2(3,3) | $R_1$ (1,1) | $R_5$ (2,10.5) | 16 | $R_1$ (1,1) |
| T3(7,10.5) | $R_4$ (10,7) | $R_5$ (2,10.5) | 16 | $R_4$ (10,7) |
| T4(10,1) | $R_1$ (3,3) | $R_4$ (7,10.5) | 18 | $R_1$ (3,3) |
| T5(3,8) | $R_5$ (2,10.5) | $R_4$ (7,10.5) | 17 | $R_5$ (2,10.5) |

## 5.3 Case III

This is an interesting case, where number of robot requirement of each target is variable. Some targets might require one robot , some might require two or, some might more than that. So, here number of robots is more than the number of targets. As input (target) comes, the output (robot) keep on assigning the task. In this particular case let there are nine numbers of robots and only five target locations. Let T1 needs only one robot, T2 needs two robots, T3 needs three robots, T4 needs two robots and T5 needs only one.

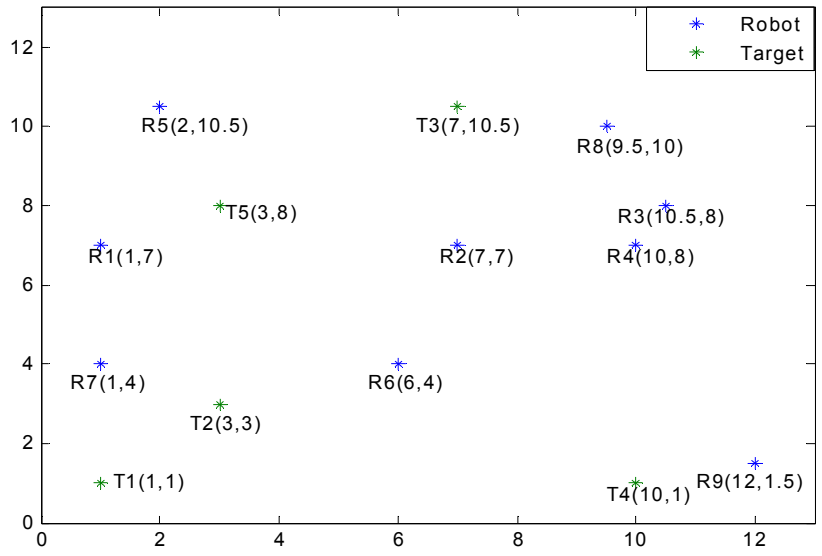


Figure 5.5: Workspace for case 3 when R>T

### Input 1

Let the first input is target location $T_1$ (1,1). Now its distance will be found from each robot and on the basis of minimum distance travel method, a particular robot will be selected. For different robots there will be different distance measurement. As shown in table 5.22 below:

Table 5.22: Distance measurement for input 1

| Robot | Distance | |
|---|---|---|
| R1 | D11 | 6 |
| R2 | D12 | 8.47 |
| R3 | D13 | 11.77 |
| R4 | D14 | 10.8 |
| R5 | D15 | 9.5 |
| R6 | D16 | 5.82 |
| R7 | D17 | 3 |
| R8 | D18 | 12.34 |
| R9 | D19 | 11.01 |

Here, minimum distance found is 6.And the distance next to it is 8.485. So, for target $T_1$, the robot $R_1$ is the winner and robot $R_2$ is the neighbour. Now the approximate path which these robots could follow will be found. Hence the path to be followed will be:

Table 5.23:Path planning for input 1

| | |
|---|---|
| R7 | (1,4)(1,2.5)(1,1.75)(1,1.375)(1,1.1875)(1,1.0938)(1,1.0469)(1,1.0234)(1,.10117)<br>(1,1.0059)(1,1.0029)(1,1.0015)(1,1.0007)(1,1.0004)(1,1.0002)(1,1.0001)(1,1) |

## Input 2

Now, next input will be given i.e. next target location is specified. The input is target 2 located at (3,3). Here in this case, two robots are required. So, we will find the distance of T2 from all rest of robots. Now, distance of target T2 from all robots is tabulated below

Table 5.24: Distance measurement for input 2

| Robot | Distance | |
|---|---|---|
| R1 | D21 | 4.47 |
| R2 | D22 | 5.64 |
| R3 | D23 | 9 |
| R4 | D24 | 8.05 |
| R5 | D25 | 7.56 |
| R6 | D26 | 3.16 |
| R8 | D28 | 9.52 |
| R9 | D29 | 9.12 |

Here two consecutive minimum distances are , That of robots R1 and R6. There Path Planning is given in table 5.24

Table 5.25:Path planning for input 2

| | |
|---|---|
| R6(6,4) | (6,4)(4.5,3.5)(3.75,3.25)(3.375,3.125)(3.1875,3.0625)(3.0938,3.0313) (3.0469,3.0156)(3.0234,3.007)(3.00117,3.0039)(3.0059,3.0020)(3.0029,3.0010) (3.0015,3.0005)(3.0007,3.0002)(3.0004,3.0001)(3.0002,3)(3.0001,3)(3,3) |
| R1(1,7) | (1,7)( 2,5)(2.5,4)(2.75,3.5)(2.875,3.25)(2.9375,3.125)(2.9688,3.0625)(2.9844, 3.0313)(2.9922,3.0156)(2.9961,3.0078)(2.9980,3.0039)(2.999,3.0020)(2.9995, 3.001)(2.9998,3.0005)(2.9999,3.0002)(2.9999,3.0001)(3,3.0001)(3,3) |

## Input 3

Next input target location is T5, and it is located at (3,8). This target location requires only one robot. Now, its distance from all rest of the five robots will be measured the robots.

Table 5.26: Distance measurement for input 3

| Robot | Distance | |
|---|---|---|
| R2 | D52 | 4.12 |
| R3 | D53 | 7.5 |
| R4 | D54 | 7.07 |
| R5 | D55 | 2.69 |
| R8 | D58 | 4.47 |
| R9 | D59 | 6.8 |

Now the winner will be selected on the basis of minimum distance method. Here for input T5, R5 is at minimum distance and R5 is next to it. Again by applying the neighbourhood function to the two robots the path will be planned.

Table 5.27:Path Planning for input 3

| | |
|---|---|
| R5(2,10.5) | (2,10.5)(2.5,9.25)(2.75,8.625)(2.875,8.3125)(2.9375,8.1563)(2.9688,8.0781)(2.9844,8.03915555)<br>(2.9922,8.0195)(2.9961,8.0098)(2.9980,8.0049)(2.999,8.0024)(2.9995,8.0012)(2.9998,8.0006<br>(2.9999,8.0003)(2.9999,8.0002)(3,8.0001)(3,8) |

## Input 4

Out of the rest of the two target location, now the input is target T3 loacted at (7,10.5).Three robots are needed by this target location. Three cosecutive minimum distances will be found an will be set as winners

.

Table 5.28: Distance measurement for input 4

| Robot | Distance | |
|---|---|---|
| R2 | D32 | 3.47 |
| R3 | D33 | 4.2 |
| R4 | D34 | 4.6 |
| R8 | D38 | 2.54 |
| R9 | D39 | 10.18 |

Table 5.28 shows that R2,R3 and R8 are at less distance from target becomes winner and. Now, let us calculate their feasible path to the target. The path to be followed is shown in Table 5.29.

Table 5.29:Path planning for input 4

| | |
|---|---|
| R2(7,7) | (7,7)(7,8.75)(7,9.625)(7,10.0625)(7,10.2813)(7,10.3906)(7,10.4453)(7,10.4727)(7,10.4863) (7,10.4932)(7,10.4966)(7,10.4983)(7,10.4991)(7,10.4996)(7,10.4998)(7,10.4999)(7,10.4999) (7,10.5) |
| R4(10,7) | (10,7)(8.5,8.75)(7.75,9.625)(7.375,10.0625)(7.1875,10.2813)(7.0938,10.3906)(7.0469,10.4453) (7.0234,10.4727)(7.0117,10.4863)(7.0059,10.4932)(7.0029,10.4966)(7.0015,10.4983)(7.0007, 10.4991)(7.0004,10.4996)(7.0002,10.4998)(7.0001,10.4999)(7,10.4999)(7,10.5) |
| R8(9.5,10) | (9.5,10)(8.25,10.25)(7.625,10.375)(7.3125,10.4375)(7.1563,10.4688)(7.0781,10.4844)(7.0391, 10.4922)(7.0195,10.4961)(7.0098,10.4980)(7.0049,10.4990)(7.0024,10.4995)(7.0012,10.4998) (7.0006,10.4999)(7.0003,10.4999)(7.0002,10.5)(7.0001,10.5)(7,10.5) |

## Input 5

This is the final and last input. This target requires two robots and only two are left in the race. So, the path to be followed by rest of these robots R4 and R9 is given in table 5.30.

Table 5.30:Path planning for input 5

| | |
|---|---|
| R3(10.5,8) | (10.5,8)(6.25,9.25)(4.125,9.875)(3.0625,10.1875)(2.5313,10.3438)(2.2656,10.4219)(2.1328, 10.4609)(2.0664,10.4805)(2.0332,10.4902)(2.0166,10.4951)(2.0083,10.4976)(2.0042,10.4988) (2.0021,10.4994)(2.001,10.4997)(2.0005,10.4998)(2.0003,10.4999)(2.0001,10.5)(2,10.5) |
| R9(12,1.5) | (12,1.5)( 7,6)(4.5,8.25)(3.25,9.375)(2.625,9.9375)(2.3125,10.2188)(2.1563,10.3594)(2.0781, 10.4297)(2.0391,10.4648)(2.0195,10.4824)(2.0098,10.4912)(2.0049,10.4956)(2.0024,10.4978) (2.0012,10.4989)(2.0006,10.4995)(2.0003,10.4997)(2.0002,10.4999)(2.0001,10.4999)(2,10.5) |

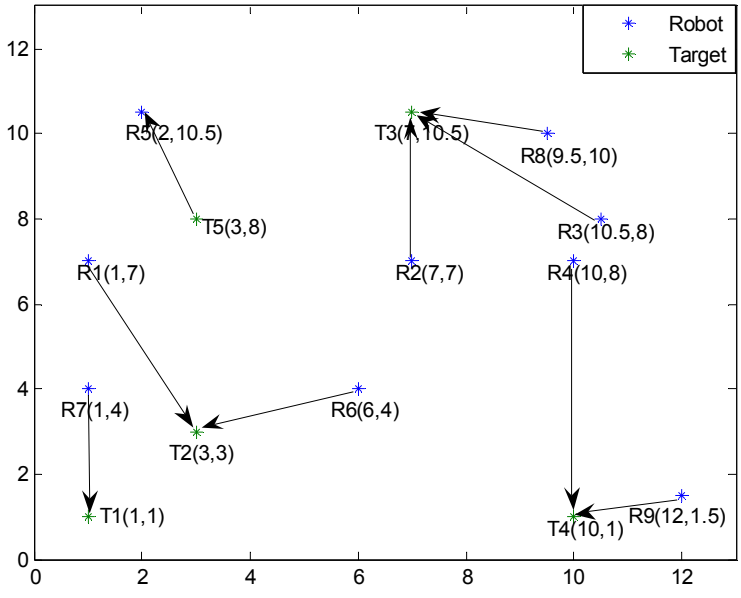


Figure 5.6: Path Planning of case 3

The all above calculations done for Case 2 (when R>T) is tabulated in Table 5.31.

Table 5.31: Path planning and task assignment of all targets for case 3

| Input(Given Target loaction) | Winner nodes |
|---|---|
| T1(1,1) | $R_7$ (1,4) |
| T2(3,3) | $R_1$(1,7),R6(6,4) |
| T3(7,10.5) | $R_2$(7,7),R3(10.5,8),R8(9.5,10) |
| T4(10,1) | $R_4$ (10,7),R9(12,1.5) |
| T5(3,8) | $R_5$ (2,10.5) |

## Case IV

This is a somewhat difficult case, where targets are also moving. Each time robot has to trace the target and according to that change its position. The targets can move at any speed and in any direction. During this study it is considered that all targets are moving in North-West direction but are moving at a speed slower than Robots.

Some other considerations have also been done:
Here like case I, R=T and are positioned in same pattern.
Target will cover a distance of 0.1 only in one step.
Target is three times slower than Robot.

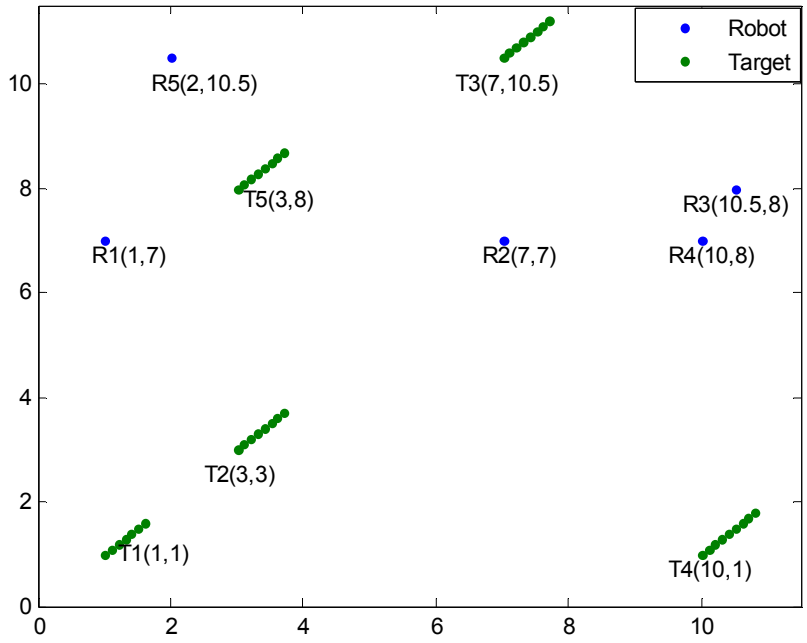


Figure 5.7: Workspace for the case when T(x1,y1)→ T(x2,y2)→ T(x3,y3)

Table 5.32: Path Planning For all inputs

| Target Movement | Robot Movement |
|---|---|
| T1(1,1)(1.1,1.1) | R1(1,7)(1,1.6)(1,1.06)(1,1.006)(1,1.0006)(1.09,1.09)(1.099,1.099)(1.0999,1.0999)(1.1,1.1) |
| T2(3,3)(3.1,3.1) | R2(7,7)( 3.4,3.4)(3.04,3.04)(3.004,3.004)(3.0004,3.0004)(3.09,3.09)(3.099,3.099)(3.0999,3.0999)(3.1,3.1) |
| T3(7,10.5)(7.1,10.6) | R3(10.5,8)(7.35,10.25)(7.035,10.475)(7.0035,10.4975)(7.0004,10.4998)(7.09,10.59)(7.099,10.599)(7.0999,10.5999)(7.1,10.6) |
| T4(10,1)(10.1,1.1) | R4(10,7)(10,1.6)(10,1.06)(10,1.006)(10,1.0006)(10.09,1.0901)(10.099,1.099)(10.0999,1.0999)(10.1,1.1) |
| T5(3,8)(3.1,8.1) | R5(2,10.5)(2.9,8.25)(2.99,8.025)(2.999,8.0025)(2.9999,8.0002)(3.09,8.09)(3.0990, 8.099)(3.0999,8.0999)(3.1,8.1) |

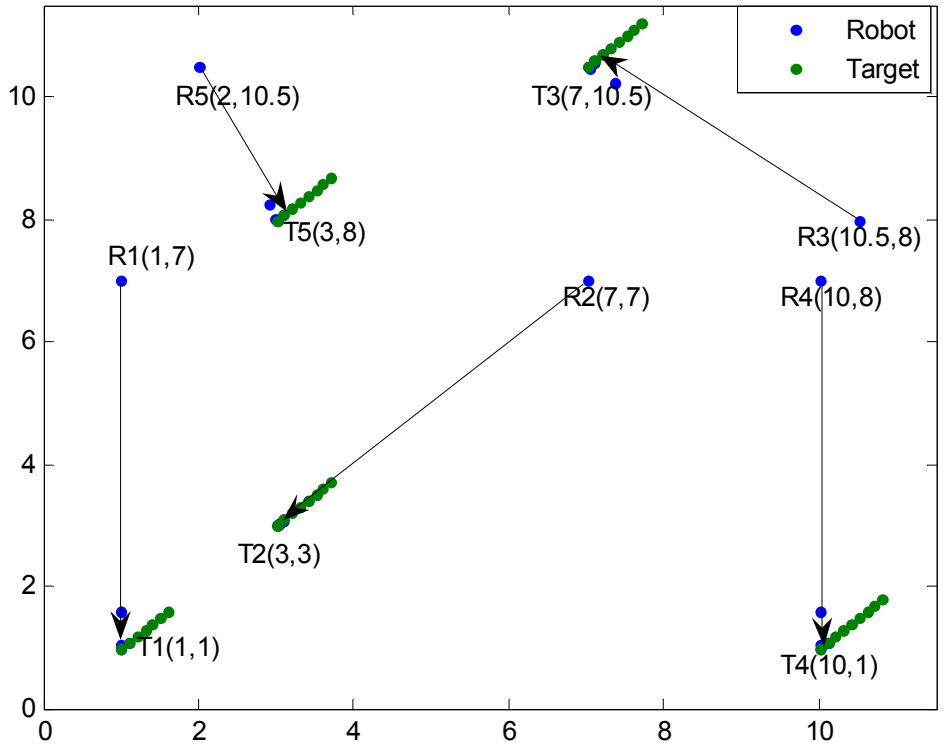


Figure 5.8: Path Planning for case IV

## 5.2 Case study II

Its about the study of, effect of change of different parameters over the spped of the system and the path planning of robots.

The values of learning rate $\chi$, have been varied from 0..05 to1 at a constant interval of 0.1. Hence different path planning will be obtained for 10 cases varying $\chi$ as 0.05, 0.06,0.07……..so on till 1.And other parameters will remain constant. Here $\gamma$ =0.03, A=1 and $\lambda$=0.2. The path Planning of Robot R1 to rach upto Target T1 is done using different learning rate. Only one case has been considered, as the result will be same for all robot movement. Table 5.33 gives the result:

Table 5.33:Iterations required at different values of $\chi$

| $\chi$ | No. of Iterations Required |
|---|---|
| 0.05 | 5 |
| 0.06 | 5 |
| 0.07 | 5 |
| 0.08 | 5 |
| 0.09 | 5 |
| 0.1 | 6 |
| 0.2 | 8 |
| 0.3 | 10 |
| 0.4 | 13 |
| 0.5 | 17 |
| 0.6 | 23 |
| 0.7 | 33 |
| 0.8 | 53 |
| 0.9 | 114 |
| 1.0 | more |

# Chapter 6

## Results and Discussion

After performing all types of simulation on all types of possible cases. It has been found that the neural network approach used in this algorithm is efficient to update the weights and hence robot location. It can easily do path planning and task assignment and hence can control the movement of robots within a workspace. It will help in collision avoidance.

The advantageous feature of this algorithm is that it combines motion planning and task assignment for a multi-mobile robot system. It can deal easily with complicated and worst (robot breakdown) cases also.

The path planning of each robot for each input and also for different cases is shown below in figure 6.1 to 6.4.

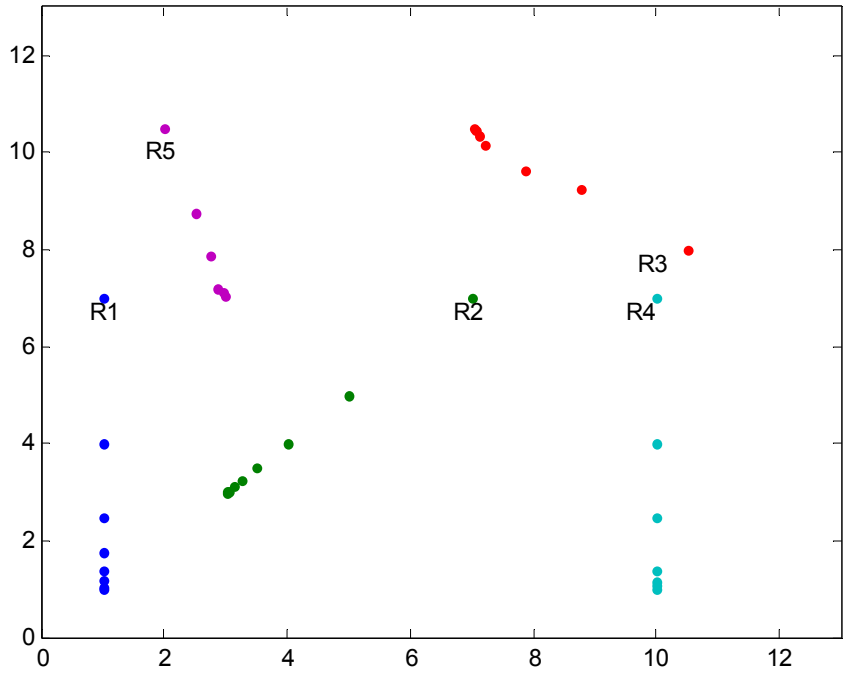


Figure 6.1: Point to Point movements for Case I

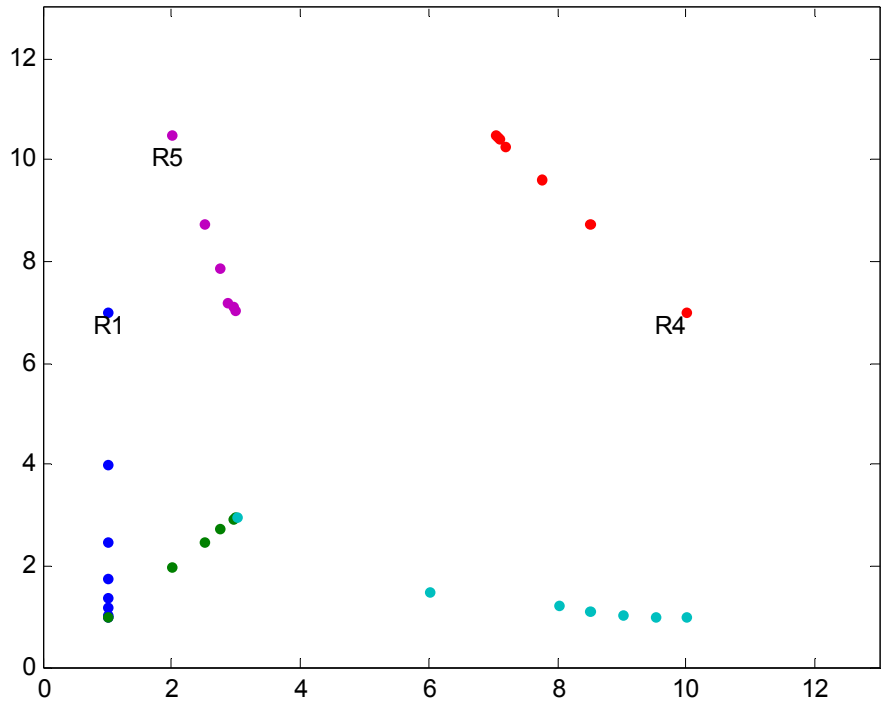


Figure 6.2: Point to Point movements for Case II

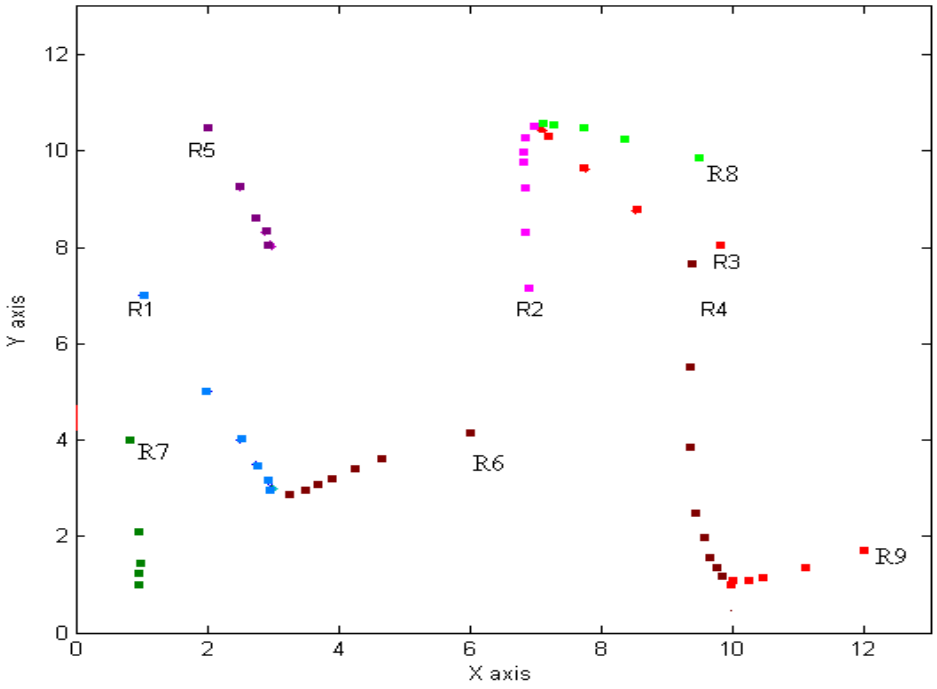


Figure 6.3: Point to Point movements for Case III

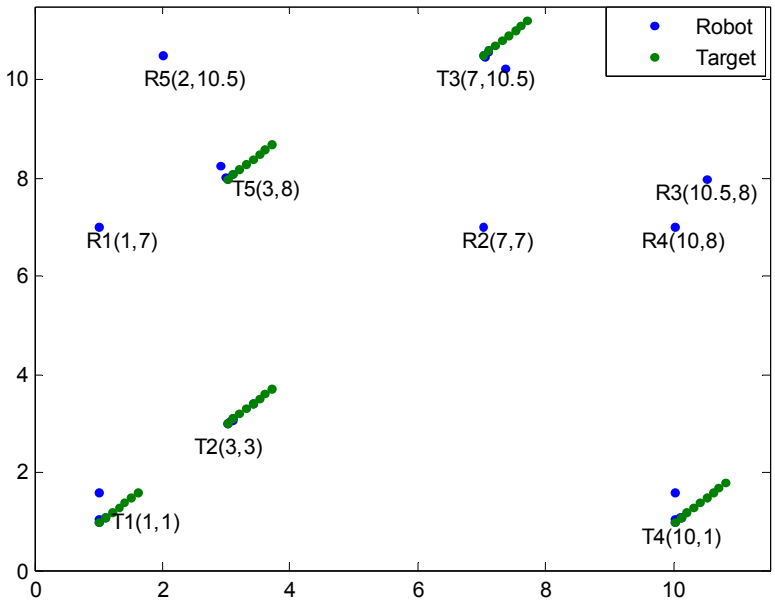


Figure 6.4: Point to Point movements for Case IV

The effect of change of learning rate has also been studied. It has been found that multi mobile robot system gives best efficiency for learning rate lying between the range (0.08 to 0.2) . The figure 6.5 below shows an optimized path planning for $\chi = 0.1$ , Considering only Case1 of Case study I.

# Conclusion & Future Scope

It has been concluded that the use of Self Organizing Feature Map of Neural Networks system in path planning of robot gives an optimized result. Till now only such algorithms like distributed auction algorithm, agent based algorithm, genetic based algorithm, dynamic tabu search algorithm have been developed which concentrates on task assignment problems only. He consequences of these algorithms were that the robot remains idle for sometime. And also they can't deal with complicated situations.

In this system the Path Planning and Task Assignment is done simultaneously hence there is no wastage of time and also it can deal with all type of rigid situations. This algorithm has relatively a simple model.

The sensitivity of learning rate is checked and it is found that it gives best results in the range 0.08 to 0.2. Before this Range it is insensitive and after this range number of iterations increases exponentially.

This algorithm helps in calculating efficiency of system. And, also using this approach the real time location of each robot can be viewed. Hence it demonstrates the live performance of every robot.

In this thesis, we have not considered the Path Planning strategy for Randomly moving targets and if targets move faster than robots. A fast algorithm should be developed for movable targets.

The sensitivity of parameter should be known to develop a robust system. Hence, the effect of other parameter changes should also be consider such as effect of change of Gain change parameter.

# References


[1] JORGE ANGELES, ANGEL ROJAS, AND CARLOS S. LOPEZ-CAJUN, "Trajectory Planning in Robotics Continuous-Path Applications" IEEE JOURNAL OF ROBOTICS AND AUTOMATION, VOL. 4, NO. 4,PP 360-365, AUGUST 1988

[2] Musa K.Jouaneh, Zhixiao Wang, and David A. Dornfeld, "Trajectory Planning for Coordinated Motion of a Robot and a Positioning Table: Part 1 –Path Specification" IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION , Vol. 6, No. 6, PP 735-745, DECEMBER 1990

[3]Zeungnam Bien and Jihong Lee, "A Minimum-Time Trajectory Planning Method for Two Robots", IEEE. TRANSACTIONS ON ROBOTICS ANI) AUTOMATION. VOL. 8, NO. 3, PP 414-418, . JUNE 1992

[4] Mitsuo Gent, Runwei Cheng', and Dingwei Wang, "Genetic Algorithms for Solving Shortest Path Problems", Evolutionary Computation, 1997., IEEE International Conference on, 13-16 Apr 1997 On page(s): 401-406.

[5]Gill, M.A.C.; Zomaya, A.Y., "A parallel collision-avoidance algorithm for robot manipulators",Concurrency, IEEE Volume 6, Issue 1, Jan.-March 1998 Page(s):68 – 78.

[6]LaValle, S.M.; Hutchinson, S.A., "Optimal motion planning for multiple robots having independent goals.",Robotics and Automation, IEEE Transactions on Volume 14, Issue 6, Dec. 1998 Page(s):912 - 925

[7]Lasky, T.A.; Ravani, B., "Sensor-based path planning and motion control for a robotic system for roadway crack sealing.",Control Systems Technology, IEEE Transactions on Volume 8, Issue 4, July 2000 Page(s):609 - 622

[8]Yongji Wang; Lane, D.M., "Solving a generalized constrained optimization problem with both logic AND and OR relationships by a mathematical transformation and its application to robot motion planning."Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on Volume 30, Issue 4, Nov. 2000 Page(s):525 - 536

[9]Hourtash, A.; Tarokh, M., "Manipulator Path Planningby decomposition: algorithm and analysis", Robotics and Automation, IEEE Transactions on, Volume 17, Issue 6, Dec. 2001 Page(s):842 - 856

[10] Dong Sun, Member, IEEE, and James K. Mills, "Adaptive Synchronized Control for Coordination of Multirobot Assembly Tasks", IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 18, NO. 4,,PP498-510, AUGUST 2002.

[11]Jiacheng Tan; Clapworthy, G.J., "Virtual environments for Internet-based robots. II. Path planning."Proceedings of the IEEE Volume 91, Issue 3, March 2003 Page(s):389-395.

[12]Ladd, A.M.; Kavraki, L.E., "Measure theoretic analysis of probabilistic path planning"
Robotics and Automation, IEEE Transactions on Volume 20, Issue 2, April 2004 Page(s):229 – 242.

[13] Xuena Qiu and Shirong Liu, Simon X. Yang, "A Rolling Method for Complete Coverage Path Planning in Uncertain Environments", Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics August 22 - 26, 2004, Shenyang, China, PP146 – 151.

[14]Ge, S.S.; Xuecheng Lai; Mamun, A.A., " Boundary following and globally convergent path planning using instant goals", Systems, Man, and Cybernetics, Part B, IEEE Transactions on Volume 35, Issue 2, Apr 2005 Page(s):240 - 254

[15] Dongkyoung Chwa, Junho Kang, and Jin Young Choi, "Online Trajectory Planning of Robot Arms for Interception of Fast Maneuvering Object Under Torque and Velocity Constraints", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 35, NO. 6, PP 831-843, NOVEMBER 2005.

[16] Kian Hsiang Low, Wee Kheng Leow, Member, IEEE, and Marcelo H. Ang, Jr., Member, IEEE, "Autonomic Mobile Sensor Network With Self-Coordinated Task Allocation and Execution", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, VOL. 36, NO. 3, PP 315- 327, MAY 2006

[17] Emmanuel A. Merchán-Cruz and Alan S. Morris, "Fuzzy-GA-Based Trajectory Planner for Robot Manipulators Sharing a Common Workspace", IEEE TRANSACTIONS ON ROBOTICS, VOL. 22, NO. 4, PP 613-624, AUGUST 2006

[18] 'Kyung-Seok Park and 2Han-Soo Choi, "Neural Network Based Path Planning Plan Design of Autonomous Mobile Robot", SICE-ICASE International Joint Conference 2006,PP3757-3761, Oct.18-21, 2006 in Bexco, Busan, Korea

[19] Harry Chia-Hung Hsu and Alan Liu, "A Flexible Architecture for Navigation Control of a Mobile Robot", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 37, NO. 3, PP310-318, MAY 2007

[20] Xiaobu Yuan and Simon X. Yang, "Multirobot-Based Nanoassembly Planning with Automated Path Generation", IEEE/ASME TRANSACTIONS ON MECHATRONICS, VOL. 12, NO. 3, PP 352- 356, JUNE 2007.

[21] Foudil, "Trajectory Generation for Mobile Manipulators Using a Learning Method", Mediterranean Conference on Control and Automation,PP 1-6, July 27-29,2007.

[22] Hairong Xiao Li Liao and Fengyu Zhou, "Mobile Robot Path Planning Based on Q-ANN", Proceedings of the IEEE, PP 2650-2654 International Conference on Automation and Logistics August 18 - 21, 2007, Jinan, China

[23] Ladaloharivola Randria, Mohamed Moncef Ben Khelifa, Moez Bouchouicha, Patrick Abellard," A Comparative Study of Six Basic Approaches for Path Planning Towards an Autonomous Navigation", The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON), PP2730-2735, Nov. 5-8, 2007, Taipei, Taiwan.

[24] Xiaobu Yuan,Simon X. Yang, "Multi-Robot Coordination with Balanced Task Allocation and Optimized Path Planning", Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics, PP1007-1011 December 15 -18, 2007, Sanya, China.

[25] Huan Tan, Qingmin Liao Jianxun Zhang, "An Improved Algorithm of Multiple Robots Cooperation in Obstacle Existing Environment", Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics, PP1001-1006 December 15 -18, 2007, Sanya, China.

[26] Meijuan Gao Jingwen Tian, "Path Planning for Mobile Robot Based on Improved Simulated Annealing Artificial Neural Network", IEEE Third International Conference on Natural Computation , PP1-5,(ICNC 2007).

[27] Shuai Li, Max Q. H. Meng, Wanming Chen, Yangming Li, Zhuhong You, Yajin Zhou 1, 2, Lei Sun 1, 2, Huawei Liang1, Kai Jiang 1, 2, Qinglei Guo1, 2, "SP-NN: A Novel Neural Network Approach for Path Planning", Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics, PP1355-1360 December 15 -18, 2007, Sanya, China

[28] Mike Peasgood, Associate Member, IEEE, Christopher Michael Clark, and John McPhee, "A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps", IEEE TRANSACTIONS ON ROBOTICS, VOL. 24, NO. 2, PP 283-292, APRIL 2008.

# Appendix-I

## MATLAB CODES

```
% Program for constructing Workspace
i=input('prompt')       % No. of Robots
j=input('prompt')       % No. of Targets
for i=1:5
x(i)=input('prompt')   %x coordinates of Robot
y(i)=input('prompt')   %y coordinates of Robot
end
for j=1:5
x1(j)=input('prompt')   %x coordinates of Target
y1(j)=input('prompt')   %y coordinates of Target
end
h=plot(x(i),y(i),x1(j),y1(j))
axis([0 11.5 0 11.5]);
set(h,'marker','.')
set(h,'linestyle','none')
%set(h,'marker','*')
h=legend('Robot','Target',2);
```

```
%Program for Calculating the distances

i=input('prompt')       % No. of Robots
j=input('prompt')       % No. of Targets
for i=1:5
Rx(i)=input('prompt')   %x coordinates of Robot
Ry(i)=input('prompt')   %y coordinates of Robot
end
for j=1:5
Tx(j)=input('prompt')   %x coordinates of Target
Ty(j)=input('prompt')   %y coordinates of Target
end
for j=1:5
   for i=1:5
dist(j,i)=distance(Rx(i),Ry(i),Tx(j),Ty(j))  %Distance of each Robot from each target
end
end
```

```
% Program for calculating Neibhorhood function
y=0.03;
A=1;
z=0.2;
x=0.2;
for i=1:20
y=y*(0.998)^i;    % Calculating gain change rate
A=A*(1-y)^i;      % Calculating Gain Parameter
for R=0:2
  d(r)=exp(-(R^2/A^2))
end
end
d(r)        % display all values of d(r)
```

```
%Program for Path Planning of Robots .
i=input('prompt')     % No. of Robots
j=input('prompt')     % No. of Targets
for i=1:5
Rx(i)=input('prompt')    %x coordinates of Robot
Ry(i)=input('prompt')    %y coordinates of Robot
end
for j=1:5
Tx(j)=input('prompt')   %x coordinates of Target
Ty(j)=input('prompt')   %y coordinates of Target
end
for j=1:5
   for i=1:5
dist(j,i)=distance(Rx(i),Ry(i),Tx(j),Ty(j))  %Distance of each Robot from each target
end
end
min.d = minimum(dist(j,i))   % Finding the minimum distance
y=0.03;
A=1;
z=0.2;
x=0.2;
for n=1:20
y=y*(0.998)^n;        % Calculating gain change rate
A=A*(1-y)^n;          % Calculating Gain Parameter
for r=0:2
  d(r)=exp(-(r^2/A^2))   % Calculating Neighborhood function
 end
```

```
if (dist(j,i)<(z*min.d)
    Rx(i) = Tx(j)
    Ry(i) = Ty(j)
end
 Rx(i)=Tx(j)+(x*d(r)*(Rx(i)-Tx(j)))   % Weight updation
Rx(i)
Ry(i)=Ty(j)+(x*d(r)*(Ry(i)-Ty(j)))
Ry(i)
End
```

# Appendix-II

## Curriculum Vitae

Apra Gupta
G 324 A, HIG Flats,
Sec 11, Pratap Vihar
Ghaziabad.
E-mail: **apra.engg@gmail.com**
Mobile No. 09878813181
09871577447

---

## Educational Qualification:

Post Graduation: 'Master of Engineering' in 'Electronics Instrumentation and Control Engineering' from Thapar University, Patiala with current C.G.P.A. 7.82

Thesis: An Efficient Algorithm for Path Planning Task Assignment of Multi-Mobile Robot System Using ANN

Graduation: 'Bachelor of Technology' in 'Applied Electronics and Instrumentation' from Uttar Pradesh Technical University (Lucknow) with 70% aggregate

V.K.V School Ghaziabad, (2001-2002) C.B.S.E. with 71% aggregate.

---

## Projects (B.Tech.)

- Working on PC based PLC controller.
- A Tell Tale Clock for Industries.

## Project(M.E.)

- Designing Full Adder using Microcontroller.

## Summer Training:

Four weeks summer training during July-Aug.2004 at Railtel Corporation of India (a Govt. of India undertaking) and gained an experience in the field of communication with latest technology.

Four weeks summer training undertaken during July-Aug .2005 at Railway Microwave Labs. (Digital and Analog), learning operation, maintenance and repair of various types of measuring instruments.

---

## Strength:

- Sincere and punctual
- Hardworking and deterministic.
- Willingness
- Leadership

---

## Computer Skills

**Operating system** : win 98, XP, MS-DOS
**Others** : Knowledge of Internet, Basics of C, C++and Data Structure.

---

## Extra Curricular Activities:

- Third prize in Long Jump 'Resonance 2k4' Organized by ABES Engg. College.
- Second prize in the Group Dance held by Organization Committee of 2k5,
- Participated in Jawahar Lal Nehru K.V.S. Regional Science Exhibition, 2001 organized at Kendriya Vidyalaya, New Delhi.

---

## References

**Dr. Y.D.Singh**, Dept of electrical and instrumentation, T.I.E.T., Patiala
**Dr. R.S.Aggarwal**, Dept of electrical and instrumentation, T.I.E.T., Patiala