DESIGN OPTIMIZATION OF FORK AND JOIN OPEN ASSEMBLY SYSTEMS VIA SIMULATION METAMODELING AND GENETIC ALGORITHMS

MAYUR NISHIKANT SHASTRI

A

Thesis

In

The Department

of

Mechanical & Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Quebec, Canada

Your file Votre référence
ISBN: 0-494-04430-6
Our file Notre référence
ISBN: 0-494-04430-6

ABSTRACT

Fork and Join Assembly Systems are increasingly being used in modern manufacturing systems. Due to the complexities in their configurations, designing such systems for an optimal performance may pose a number of challenges. Because of the involvement of large system parameters and variables, designing an assembly system is not an easy task. This study presents a design optimization approach in Fork and Join Open Assembly Systems. Optimal buffer allocations to accommodate the work-in-process inventories in such systems are optimized in an attempt to maximize the overall system production rate. Fundamentally the problem is a stochastic, nonlinear, combinatorial optimization problem with discrete decision variables. Because of the nature of the problem, it is extremely difficult to find any closed-form expression to determine the expected value of the production rate. Hence discrete event simulation is used to estimate the expected value of the production rate. Then simulation model coupled with genetic algorithms is used to find optimal buffer configuration for maximum production rate. Results obtained proved the efficiency of simulation optimization method. However, simulation is extremely time consuming due to lengthy computational requirements for most of the real life problems. Especially, if system has large parameter space, then any parametric study becomes impractical. Hence we decided to employ a novel approach of simulation metamodeling. Simulation metamodels are the models of the simulation models and are much simpler form of real systems. It provides flexibility in terms of parametric study if systems have large parameter space. In this study, feedforward, backpropogation based Artificial Neural Network (ANN) is developed for the system under consideration. ANN together with GA is then used to find optimal buffer configuration for maximum production rate. We compared SGA approach with ANN-GA approach and concluded that ANN-GA gives comparable solutions with more flexibility and less computational time, which eventually helps systems design engineers to take quick decisions regarding production controls.

“Vakratund Mahakay Surya Koti Samaprabha

Nirvighnam kurme Devaha Sarva Karyeshu Sarvada”

ACKNOWLEDGEMENT

The author is extremely thankful to his supervisor Dr. A. A. Bulgak for his intellectual guidance and moral support at key moments.

The author would also like to thank the members of the faculty, staff and fellow students of the Department of Mechanical and Industrial Engineering, Concordia University for their assistance throughout the course of the work.

The author is extremely grateful to author's family members for their affection and continuous encouragement without which this would not have been possible.

Table of Contents

List of Figures

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Assembly Systems: Background

Assembly work has a long historic record. People from centuries, knew the creation of useful objects composed of multiple parts. However, in modern times the main objective of assembly processes is to manufacture high quality and low cost products. Many important ideas have been developed to enhance assembly processes. Assembly parts are standardized. Parts of the same type must have the same specifications. This ensures uniform quality of the parts. Parts from different sources are then assembled to get final product [1].

The division of the assembly into sub assemblies is another important innovation in assembly manufacturing. If too many parts are involved in the assembly task, then the work may be sub divided into a number of smaller tasks. Each task processes a part of the assembly. By gradually adding parts to an assembly, a finished product is obtained. Expertise can be developed in a short period because of relatively limited content of each task. Thus, assembly speed may be increased and quality can be improved. In many assembly systems, non-assembly operations such as, parts preparation, inspection, and testing operations may be inducted in order to assure the high product quality level, and facilitate assembly work [1].

1.2 Basic Terms of Assembly System

Assembly: Assembly is the process by which various parts are collected and fitted together in order to manufacture finished product. It is characterized by the parts used and the operations necessary to combine them. Parts maybe subdivided into two subgroups, components (purchased) and sub-assemblies (intermediate goods). The unfinished units of products are called work pieces.

Operation: An operation or task is the portion of total work content in an assembly system. The time required to perform an operation is called operation time.

Work Station: A workstation is a place in assembly line where a number of operations are performed on raw parts. These stations can be divided as manual or automated assembly stations depending on the subjects performing work. The set of task assigned to a station is called station load. The time required to perform allotted task/work is called station time.

Cycle Time: The cycle time is the total amount of time that work piece spends in a production line. The cycle time should be greater than largest station time. The reciprocal of cycle time is output rate or production rate of the line.

Inter-stage Buffers: Inter-stage buffer is a storage place between two successive work stations, which can hold semi finished parts when succeeding station is still busy in processing earlier part. The parts in this storage place or buffers are called as work in process inventories (**WIP**). **Blocking** and **Starvation** are the two phenomena associated with inter-stage buffers. Because of limited buffer spaces, a station may be **blocked** when the following buffer is full. The blocked station is idle until the succeeding station requires a part stored in the buffer. The station is **starved** when the input buffer of a station is empty after terminating current job. The workstation is idle until a part enters the input buffer [2].

1.3 Classification of Assembly Systems

Assembly systems are classified as closed or open loop assembly systems, and as synchronous or asynchronous assembly systems. In synchronous assembly systems, part transfer between all stations occurs simultaneously at fixed interval of time and the whole system is placed by the speed of transfer mechanism. In asynchronous assembly systems **(AASs)**, parts can be transferred independently and can be queued in front of workstation. This provides certain amount of flexibility from workstation to workstation, which results in increase in production improvement. [3]

Further, asynchronous assembly systems are built according to system requirements. AASs can be arranged in either closed loop or open loop fashion. Some assembly systems may contain serial workstations or some may contain parallel workstations. Some AAS are designed to incorporate retention of defective parts for rework. These types of AASs arranged either with feedforward or feedback loops. In AASs transfer of parts between stations handled by either transfer chain or conveyors or automated guided vehicles (AGVs). Assemblies are often transported on work carriers or pallets with fixtures that hold assemblies. A fixed number of pallets always circulate in the AASs as per system design requirements. [3, 4, 5]

1.4 Failure and Breakdowns in Assembly Systems.

Asynchronous assembly systems are high-speed production lines. Failure of a single station may cause the complete stoppage of the entire assembly systems. Apart from these regular failures, AASs are subject to station jams. Although, the station jam is a rare event, it may affect the overall system performance. A station jam may occur due to assembling of defective part at particular station or accidental drooping of part held by robotic assembly station. These jam occurrences are random events. Also, when jam occurs, operators require random amount of time to clear the jam. **Blocking** and **Starvation** are the two important consequences associated with Jam. Suppose a jam occurs at assembly station, station will stop processing assemblies for certain time until jam gets cleared by the operator. During this temporary stoppage of line, buffer space between the station experiencing jam and one or more upstream stations may get filled. The upstream station then unable to release part in buffer space or the upstream station is **blocked** from releasing the finished assemblies to be processed in the next stations. Similarly because of limited number of pallets, the jamming of station may lead to **starvation** of downstream stations. [3, 4, 5]

1.5 Research Issues in the Design of Assembly Systems.

Designing an assembly system is a complex task because of the involvement of large system parameter and variables. There are numerous approaches to this difficult task of designing assembly system. Generally, the overall assembly system design problem is decomposed into sub-problems of manageable complexity, which in turn are solved distinctively (Figure 1.1). Valid assumptions are being made to simplify the complex problems. Then the simplified problem is solved to find out the optimal solution. But because of NP-HARD (The time required to reach optimal solution increases exponentially as the size of problem increases linearly) nature, it is extremely difficult to develop any algorithm or analytical solution, which could solve all the problems of designing optimal assembly systems.

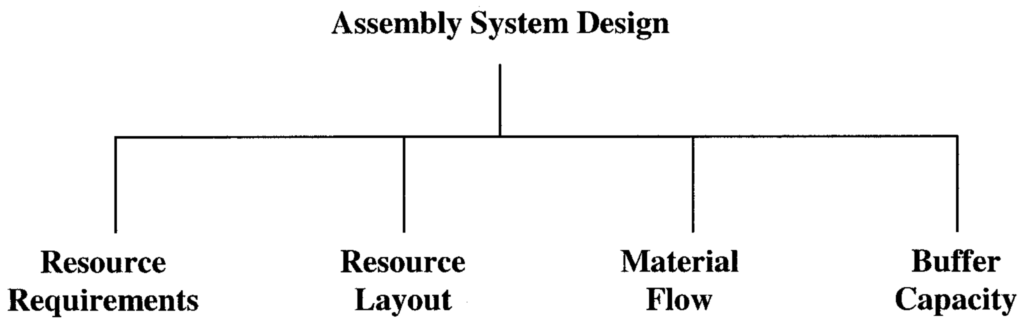


Figure 1.1 Research Issues in the Design of Assembly Systems

Resource Requirement Problem: The main task is to determine the appropriate quantity of production resource (example, machines or pallets) in manufacturing/assembly systems. Generally, the objective is cost-based, such as the maximization of investment efficiency or time-based such as the maximization of production rate.

Resource Layout Problem: This problem is related with the allocation of a set of resources in a constrained floor space. The objective is to minimize some combination of material handling cost, travel time and resource relocation cost.

Material Flow Problem: The objective is the selection of proper material handling system for assembly line so that cost is minimized, and production rate, flexibility and reliability of assembly system are maximized.

Buffer Capacity Problem: This problem deals with the optimal allocation of work-in-process (WIP) or storage capacity in assembly system. Machine utilization and production rate can be maximized with adequate levels of WIP. So it is imperative that inter stage buffer capacities be optimized in order to maximize production rate [6].

1.6 Objective of this Research.

While resource requirement problem, resource layout problem and material flow problem are out of scope of this research, this study concentrates on the inter-stage buffer capacity problem in a particular AAS topology while considering some design parameters such as jam rates, jam clear times and total number of pallets. Selection of appropriate buffer sizes in an automated manufacturing system is a complex task, subject to arbitrary fluctuations in production rate and transportation delays that are part of material handling system. If large buffer sizes are provided, then the distance between to work stations will increase which results in exceptional part transfer delays and also for large buffer sizes more WIP inventories must be provided, which results in increase in inventory cost. On the other hand, if small buffer sizes are provided, then small processing delays or station jam will result in filling of the buffer and the upstream workstations will be blocked from releasing processed part. In automated manufacturing system, for a fixed number of pallets, there is always an optimal buffer configuration capable of yielding maximum production rate with the reduction in blocking and starvation effects [3,4,5].

The problem is stochastic, nonlinear combinatorial optimization type with discrete decision variables. To this date, there is no analytical solution available to solve the complete design problem of manufacturing system, discrete event simulation is generally used to estimate production rate. Simulation has been universally used by the manufacturing systems design engineers as a flexible tool in modeling and analysis of complex manufacturing systems. It reduces cost, time and risks associated with the implementations of new designs. However, simulation is extremely time consuming due to lengthy computational requirements for most of the real life problems. If a system has large parameter space, then simulation can be impractical for the parametric study of the system performance [7]. Systematic performance studies of most real world problems are beyond reach, even with supercomputers, unless substantial improvement in the speed of the performance evaluation process can be achieved. One approach to overcome this limitation is to develop a simpler model to explain the relationship between the inputs and outputs of the system. Metamodels are the models of simulation models are increasingly being used in conjunction with the original simulation, in an attempt to improve the analysis and understanding of decision-making processes [8]. A simulation metamodel is a simpler model of the real system. The simulation model is an abstraction of the real system in which a selected subset of inputs is considered. The effect of the excluded inputs is represented in the model in the form of the randomness to which the system is subject. Simulation is used to generate data sets, which in turn are used to build the metamodel. A simulation metamodel with neural networks is a neural network whose training is provided by a simulation model. In general, a metamodel takes a fewer number of inputs and is usually simpler than the simulation model [8].

1.7 Contribution of This Research.

This research focuses on the optimal buffer allocation in fork and join open assembly system. The study has its foundation in the earlier studies of stochastic design optimization of AASs [3, 4, 5, 7, 12, 13]. Whereas, the former studies focused on various forms of closed loop AASs, we further expand the implementation domain of these technologies to the new AAS topologies which are not widely studied before. These systems are called as fork and join open assembly systems (Figure 5.2.1). Fork and join open assembly systems started gaining importance since past few years. The main areas of application are refrigeration plants and automotive production plants. The major contributions of this research are summarized as follows:

- Discrete event simulation model for fork and join open assembly system is developed from scratch. Simulation model is then coupled with Genetic

Algorithm to find optimal / near optimal buffer allocation for maximizing the production rate.

- Then, after extensive experimental investigation, an artificial neural network (ANN) metamodel is developed for simulation model of a fork and join open assembly system.
- ANN metamodel together with Genetic Algorithm is used to optimize the buffer sizes in fork and join open assembly system.

1.8 Organization of the Document.

The organization of document is as follows:

- Chapter 2 presents literature review on the study of manufacturing and assembly systems.
- Chapter 3 presents an introduction to artificial neural networks.
- Chapter 4 presents definitions of search engines and general overview of genetic algorithm and its applications in various fields.
- Chapter 5 presents design optimization problem of fork and join open assembly system, ANN model development and application of genetic algorithm as an optimizer.
- Conclusion and directions of future research are given in chapter 6.

CHAPTER 2
LITERATURE REVIEW

There are numerous studies in the area of simulation optimization of manufacturing systems. Simulation metamodelling however, started gaining importance in the last decade. Consequently literature available on simulation metamodelling is limited. The main focus of this literature review is on the allocation of buffer sizes in the assembly systems, other areas such as balancing and scheduling of assembly systems will not be provided. This review is presented in three parts. The first part will focus on design and analysis of assembly systems. The second part will be on the application of artificial neural networks in manufacturing systems and the third part is dedicated to research in the application of genetic algorithms to manufacturing systems.

2.1 Modeling, Design and Analysis of Manufacturing and Assembly Systems.

Bulgak and Sanders [3] first proposed an analytical model for the performance evaluation of on automated assembly system considering the splitting and merging of the flow of the assemblies among the main and repair loops of the assembly system. They estimated the system throughput as a function of process quality. In further Research, Bulgak and Sanders [4] presented the concept of implementation of hybrid procedures involving the use of analytical performance evaluation techniques, discreet event simulation and Monte Carlo optimization methods for stochastic design optimization of asynchronous flexible assembly systems with statistical process control (SPC) and repair loops. Dolgui and Ofitserov [9] presented new approach based on the use of discrete modification of ψ transforms jointly with some heuristics for local optimization. Their main focus was on optimization of the launching of the parts in production systems of job shop type. kouikoglou and Phillis [10] studied continuous flow model for production networks with finite buffers, unreliable machines and multiple products. They considered both acyclic and non-acyclic networks. Kavusturucu and Gupta [11] presented a methodology for analyzing finite buffer tandem manufacturing systems with N-policy. They calculated the throughput of the system using decomposition, isolation and expansion and compared their results with the simulation results. Papadopoulos and Vidalis [12] studied the optimal buffer allocation problem in short **μ-** balanced unreliable production lines. In their analysis , they presented the effect of distribution of the service and repair times , the availability of the stations and the repair rates on the optimal buffer allocation and the throughput of the lines. Bulgak et al. [13] proposed new analytical approach for designing asynchronous flexible assembly systems based on the robust design methodology, aims at studying and reducing the effect of uncontrollable factors in the process of identifying the most appropriate configuration for an AFAS. Jeong et al. [14] proposed a method for finding minimum cost configuration, which gives, desired throughput for an assembly system. They defined the configuration by the machines to be used and the buffer capacities. They proposed three heuristics, which simultaneously select the machine to be used in stations and determine the capacities of buffers. Hann and Park [15] presented an approximation method for the analysis of average steady state throughput of serial production lines with unreliable machines. They used Taylor series expansion and probability generating technique in their analysis. They proposed analytical method for optimal buffer allocation to achieve desired throughput. Paik et al. [16] proposed effective throughput approximation methods for finite buffered closed loop production systems with unreliable machines and exponentially distributed processing times. The approximation methods are based on decomposition and aggregation principles. Graupner et al. [17] presented novel approach for configuration, simulation and animation of manufacturing systems via Internet. It allows to present, test and optimizes manufacturing systems via Internet. Hemchandra and Eedupuganti [18] presented an approach for enumerating the state space and obtaining the steady state probabilities of the same for such model under exponential assumptions. They considered the finite capacity fork and join queuing model for open assembly systems with arrival and departure synchronizations for their analysis. Inman et al. [19] presented overview of research issues in the designing of production systems with respect to quality. They briefly reviewed the limited literature on the intersection of quality and production system design and suggested several new research issues that are important to industry.

2.2 Application of Artificial Neural Networks in Manufacturing and Assembly Systems.

Emelyanov and Iassinoski [20] studied an AI based object oriented tool for discrete manufacturing systems simulation. Their approach was based on creating formalization method on the basis of Artificial Intelligence (AI) and the object oriented approach. Zheng et al. [21] studied the neural network approach to the early cost estimation of packaging products. They developed the back propagation neural network model for cost estimation based on design information only. They established the correlation between costs related features and the final cost of the product by training a back propagation network using historical data. The testing results based on this approach showed good product cost estimation as compare to traditional cost estimation approaches. Kilmer et al. [22] studied the use of supervised neural networks as a metamodeling technique for the stochastic simulation of inventory model. Their results showed that, neural network metamodel is quite competitive in accuracy when compared to simulation itself and once trained can operate in nearly real time. Lee and Shaw [23] applied the neural net approach to real time flow shop sequencing. They developed two level neural networks that incrementally learn sequencing knowledge. Based on the knowledge gained from learning using a set of training examples, the real network makes real time sequencing decisions for a set of jobs that arrive in different combinations. Haouani et al. [24] studied neural network implementation for modeling and control design of manufacturing systems. Park et al. [25] studied the neural network approach along with heuristic rules to scheduling jobs on parallel machines. Dengiz and Akbay [26] use a regression metamodel to optimize batch sizes in a real printed circuit board assembly line considering a JIT model. Chen and Yang [27] designed a manufacturing system by a hybrid approach with neural network metamodelling and stochastic local search. They used back propagation neural network to generate metamodels for simulated manufacturing systems. Then they solved the optimization model by applying Simulated Annealing (SA) approach to obtain the optimal configuration with respect to objective of the systems design. Sabuncuoglu and Touhami [8] did an experimental investigation of simulation metamodeling with neural networks and illustrated that simulation metamodels with neural networks can be effectively used to estimate the system performances. Bulgak et al. [7] studied the optimization of buffer sizes in assembly systems using ANN metamodel. They developed artificial neural network metamodel for simulation of an asynchronous assembly system and ANN metamodel together with simulated annealing (SA) was used to optimize the buffer sizes in the system. Jang et al. [28] developed methodology using an artificial neural network to identify non random variation patterns to improve dimensional quality in automotive assembly process. Ghaziri and Osman [29] developed neural network algorithm for traveling salesman problem with backhauls. The major innovation of their heuristic is based on new network architecture, which consists of two separate chains of neurons. Their algorithm shows promising results in terms of solution quality and computational requirements.

2.3 Application of Genetic Algorithms in Manufacturing and Assembly Systems.

Bulgak et al. [5] applied genetic algorithms for the design optimization of asynchronous automated assembly systems. In their study, they extended the domain of application of Genetic Algorithms to Monte Carlo optimization of complex manufacturing systems. Carson [30] presented the methods and applications of simulation optimization. They mentioned methods such as Perturbation Analysis (PA), Frequency Domain method (FDM) stochastic optimization, Genetic Algorithms and Simulated Annealing. Spinellis and Papadopoulos [31] studied the stochastic algorithms for buffer allocation in reliable production lines. They compared two stochastic approaches for solving buffer allocation problem in large reliable production line. The allocation plan was calculated subject to given amount of total buffer slots using simulated annealing and genetic algorithms. Sabuncuoglu et al. [32] used Genetic Algorithms for assembly line balancing. They proposed a heuristic with special chromosome structure that is partitioned dynamically through the evolution process. Elitism is also implemented in the model by using some concepts of Simulated Annealing. Loh et al. [33] presented a genetic algorithm for printed circuit board assembly, which simultaneously solves feeder assignment and component sequencing problem. The algorithm uses unique gene selection process that increases convergence rate without degrading the quality of solution. Zhou et al. [34] presented the genetic algorithm approach for the balanced allocation of customers to multiple distribution centers in supply chain networks. Su and Chiang [35] applied the integrated approach of neural networks and genetic algorithms to optimize IC wire bonding process. Cochran et al. [36] proposed the two stage multi population genetic algorithm (MPGA) to solve parallel machine scheduling problems with multiple objectives. Choi et al.[37] presented genetic algorithm to solve asymmetric traveling sales man problem. The genetic algorithm proposed in this study extends the search space by generating and including infeasible solutions in the population. Yokoyama and Lewis [38] applied the genetic algorithm for the optimization of stochastic dynamic production cycling problem.

2.3 Concluding Remarks

The literature review presented above shows the application of intelligent techniques to various complex problems. We have summarized our observations from the literature review as follows.

- Promising results are being obtained by intelligent techniques such as ANN, GA, Simulated Annealing (SA) and Tabu Search.
- Research interest is increasing in exploring the possibility of applying intelligent techniques to different application areas.
- Domains of applications using metamodeling and metaheuristic techniques are still very limited.

In view of our analysis of the review of existing literature, we feel that it is worthwhile to expand the application domains of these promising techniques to other areas. Our objective is to implement these techniques to the design optimization problems of open assembly systems.

CHAPTER 3
ARTIFICIAL NEURAL NETWORKS: AN OVERVIEW

3.1 Introduction

Artificial Neural Networks (ANN) refers to the computing system whose central theme is borrowed from the analogy of **biological neural networks**. In other words biological brain is the basis for ANN. It is a system loosely modeled on the human brain. The field goes by many names, such as connectionism, parallel distributed processing, neuro-computing, natural intelligent systems, machine learning algorithms, and artificial neural networks. It is an attempt to simulate within specialized hardware or sophisticated software, the multiple layers of simple processing elements called neurons. Each neuron is linked to certain of its neighbors with varying coefficients of connectivity that represent the strengths of these connections. Learning is accomplished by adjusting these strengths in order to obtain appropriate results.

3.2 Structure of Biological Neuron and Artificial Neuron

In this section we will first explain what the biological neuron is and then will discuss about the structure of an artificial neuron.

3.2.1 Biological Neuron

A typical biological neuron composed of a cell body, a tubular axon, and a number of hairs like dendrites. The dendrites are extensions of a neuron which connect to other neurons to form a neural network. The axon is a long, thin tube that splits into branches terminating in little end bulbs that almost touch the dendrites of other cells. The small gap between an end bulb and a dendrite is called synapse, across which information is propagated. Basically synapses are a gateway, which connects to dendrites that come from other neurons. Hence the biological neuron can be connected to other neurons and can accept the connection from other neurons and thus form a neural network [39].

A neuron receives information from other neurons through the connections, processes it and then transmits this information to other neurons. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then output the final result. The figure 3.1 below shows a simplified biological neuron and the relationship of its four components [40].

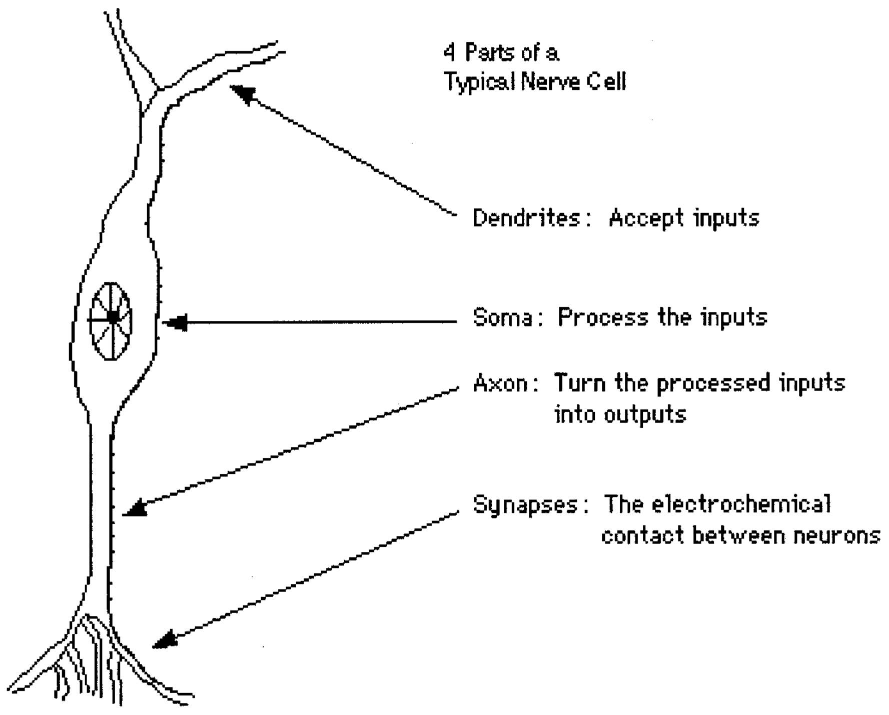

4 Parts of a
Typical Nerve Cell
Dendrites: Accept inputs
Soma: Process the inputs
Axon: Turn the processed inputs into outputs
Synapses: The electrochemical contact between neurons

Figure 3.1 Structure of a Biological Neuron

3.2.2 Artificial Neuron

The artificial neuron was originally proposed by McCulloch and Pitts [41]. It is the basic building block of the artificial neural network, simulating a biological neuron. In ANN terminology, biological neuron is termed as node or unit or cell or neurode. Other biological terms and its ANN terminology is given in table 3.1[42].

Table 3.1 Biological Terminology and Artificial Neural Network Terminology

Biological Terminology	**Artificial Neural Network Terminology**
Neuron	Node/Unit/Cell/Neurode
Synapse	Connection/Edge/Link
Synaptic Efficiency	Connection Strength/Weight
Firing Frequency	Node Output

A typical artificial neuron is given in figure 3.2 [43]. Each node receives input from some other nodes or from some external source. Each input has an associated weight *w*, which can be modified so as to model synaptic learning. The neuron sums up the net input and applies an output activation function according to the equation 3.1. Its output, in turn, can serve as input to other units. The weighted sum is called the net input to unit *i*, often referred as net_i. W_{ij} refers to the weight from unit *j* to unit *i*. The function *f* is the unit's activation function. In the simplest case, *f* is the identity function, and the unit's output is just its net input [17]. Detail explanation of output functions is given in Section 3.3.

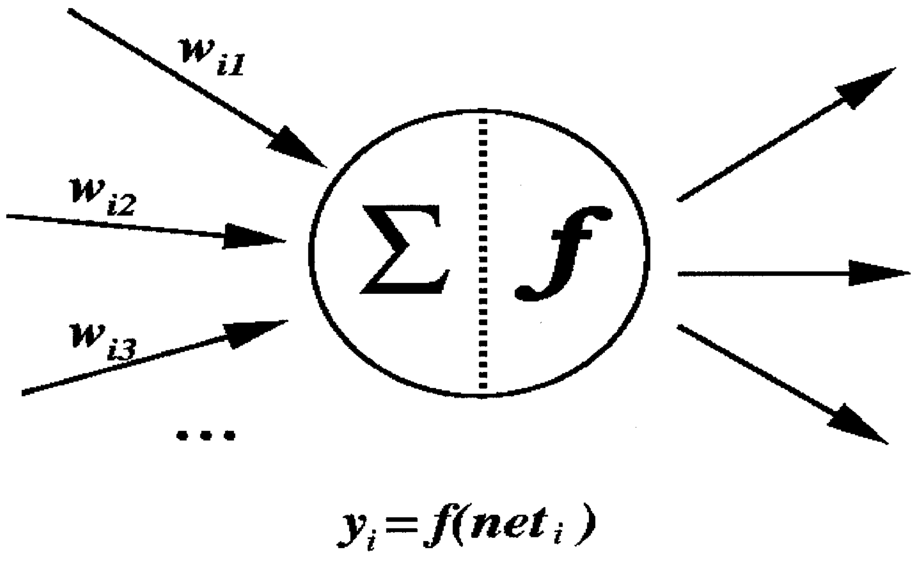


Figure 3.2 Structure of an Artificial Neuron

$$y_i = f(\sum_j w_{ij} y_j) \tag{3.1}$$

An artificial neuron is an abstract model of the biological neuron. The strength of a connection is coded in the weight. The intensity of the input signal is modeled by using a real number instead of a temporal summation of spikes. The artificial neuron works in discrete time steps; the inputs are read and processed at one moment in time[16].

3.3 Commonly Used Functions of Artificial Networks

In this Section commonly used functions of artificial networks are explained in brief. Most common functions are step, ramp, sigmoid, piecewise linear function Gaussian (radial basis function). Each is described as follows:

3.3.1 Step Function

A commonly used step function is shown in figure 3.3. The main feature of the step function is that its output does not increase or decrease to values whose magnitude is excessively high. This is desirable as biological or electronic hardware hardly produces excessively high voltages. The function defined in general as follows:

$$F\,(net) = \begin{cases} 0 \text{ if } net <= 0 \\ 1 \text{ if } net > 0 \end{cases} \qquad \{3.2\}$$

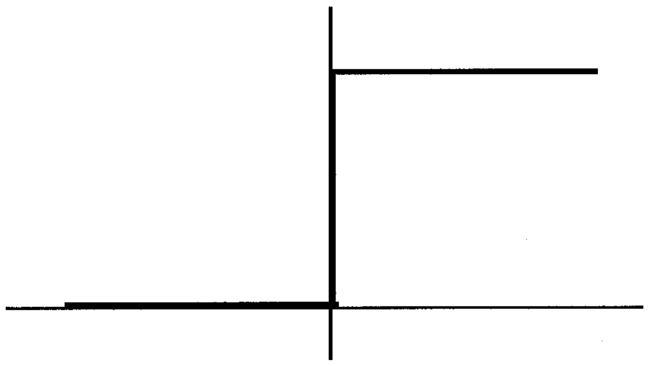


Figure 3.3 Step Function

Sigmoid (S-shaped) functions are the most popular functions used in neural net. The purpose of sigmoid function within an artificial neuron is to generate a degree of non-linearity between the neuron's input and output. Basic shape of sigmoid function is given in figure 3.4. The sigmoid function can be defined by equation 3.3. Sigmoid functions are continuous and differentiable. The main advantage of sigmoid function is that their

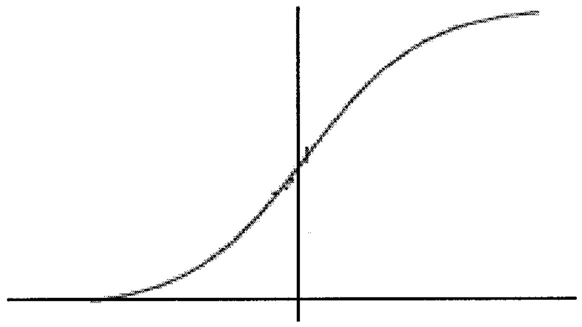


Figure 3.4 Sigmoid Function

smoothness makes it easy to device learning algorithms and understand the behavior of large networks whose node computes such functions. Also experimental observations of biological neurons illustrate that neuronal firing rate is roughly of sigmoid shape [42].

$$f(x) = \frac{1}{1+e^{-\beta x}} \qquad \{3.3\}$$

3.3.3 Piecewise Linear Function

Piecewise linear transfer function is combination s of various linear functions, where the choice of linear function depends on the relevant region of the input space. Step and ramp functions are special cases of piecewise linear functions that consists of some finite number of linear segments, and are thus differentiable almost everywhere with the second derivative equal to 0. Piecewise functions are easy to compute as compare to sigmoid functions [42]. Typical piecewise linear function is shown in figure 3.5.

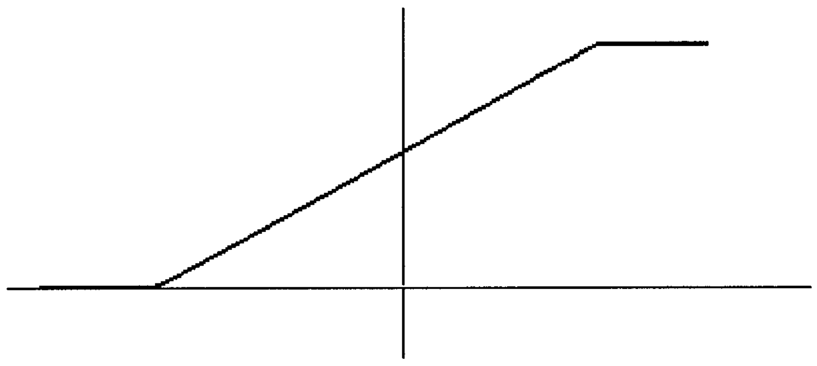


Figure 3.5 Piecewise Linear Function

3.3.4 Gaussian Functions

Gaussian functions have bell shaped curves as shown in figure 3.6. It is also known as radial basis function.

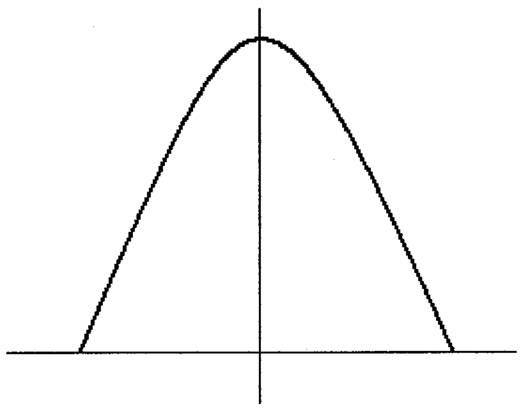
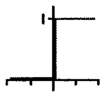
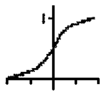
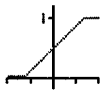
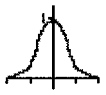
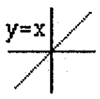


Figure 3.6 Gaussian Function

Different types of activation functions and their basic curves are summarized in table 3.2

Table 3.2 Different Functions of ANN

Unit Step	
Sigmoid	
Piecewise Linear	
Gaussian	
Identity	

3.4 Neural Net Topologies

In this Section, a detailed description of various neural network architectures is given. Single neuron is unable to solve practical problems of interest, so networks' consisting of large number of nodes connected to each other are used to solve complex problems. The manner in which these nodes are interconnected is very important since it determines how the network will perform, its processing speed and the accuracy. Hence it's very important for network designer to decide the type of network architecture to be used in early design phase.

3.4.1 Fully Connected Networks

In this type of network each node is connected to every other node. This is the most generalized type of architecture. The connections between these nodes can be excitatory (positive weights), inhibitory (negative weights) or almost zero. Fully connected network is the most general and conceptually simple type of network. Despite of its simplicity, this network is rarely used because of the large number of parameters. For instance, if network has n nodes, then total number of weights will be n^2 [42, 43]. Hence it's very difficult to devise fast learning algorithm for such a complex network. Also in biological network, it's almost impossible to establish contacts with geographically distant neurons. All other architectures or topologies of neural networks are derived from this basic architecture. In other words all other architectures are special cases of fully connected

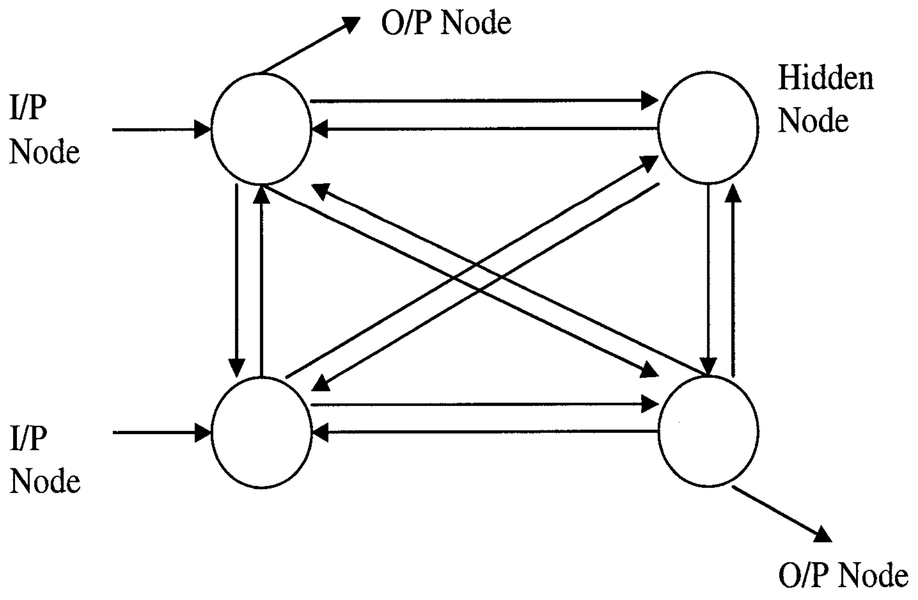


Figure 3.6 Fully Connected Network

3.4.2 Layered Networks

In this type of network, nodes are arranged in different subsets or layers. Intra-layer connections are permitted in this type of networks. The layers are divided in to three types: input layer, hidden layers and output layer. Input layer and output layer must be single layers; while there can be one or more hidden layers. Figure 3.7 shows the general arrangement of layered neurons [44]. In this type of network, neurons of the same layer may or may not be connected to each other but the neurons of one layer must always be connected to at least one other layer. Multi layered networks can solve the classification problem for non-linear sets by employing hidden layers also hidden layers enhance the separation capacity of the network [43].

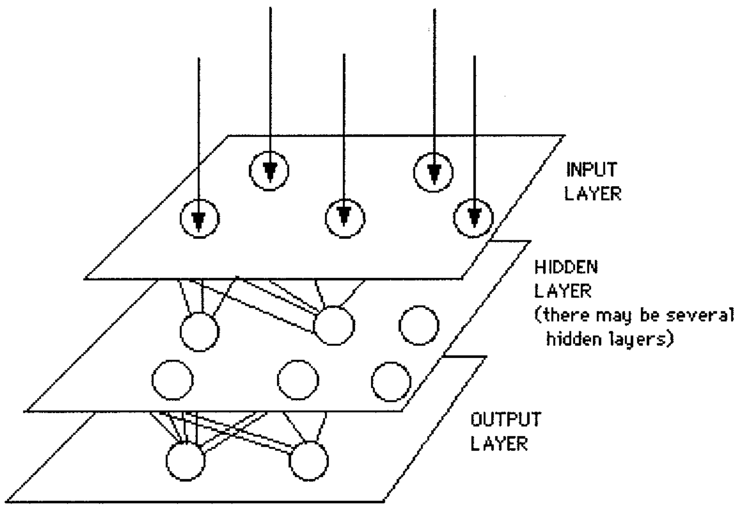


Figure 3.7 Layered Neural Network

3.4.3 Acyclic Neural Networks

In these types of networks, there are no intra-layer connections but the inter-layer connections as shown in figure 3.8. Acyclic neural network is the subclass of layered networks. In acyclic network, computational processes are much simpler than layered networks due to absence of intra-layer connections. Networks, which are not acyclic, are referred as recurrent networks.

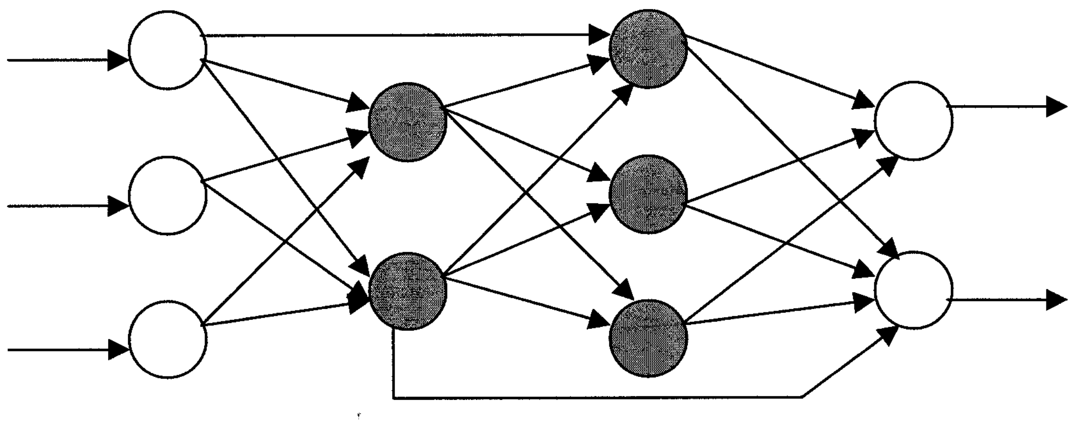


Figure 3.8 Acyclic Neural Network

3.4.4 Feedforward Neural Networks

Feedforward networks are the most popular and widely used neural networks in numerous applications. They are also known as multi-layer perceptrons. In these types of networks, only the connection from a node in layer ***n*** to other nodes in layer ***n* + 1** is allowed. These networks are described by a sequence defining the number of nodes in each layer. For example, the network shown in figure 3.8 is a 3-2-2-3 feedforward network: it contains 3 nodes in input layer, 2 nodes in first hidden layer, and 2 nodes in second hidden layer and 3 nodes in output layer [42, 45]. .

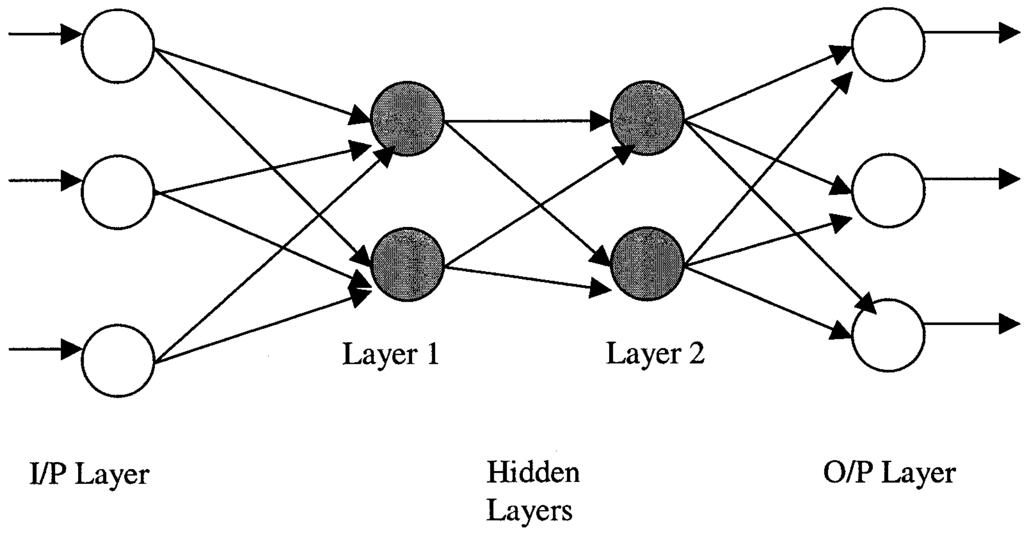


Figure 3.9 Feedforward Neural Network

3.5 Learning in Artificial Neural Networks

Learning in ANN involves with the adjustment of weights in the network so that; a set of inputs can produce desired outputs. The weights are serially adjusted according to some fixed procedures, so that the outputs progressively converge to the desired values. Learning is the vital property of neural network, which helps neural network to learn the desired response from a set of examples in contrast with other computing approaches, which require algorithms or rules to store knowledge. The benefit of learning from examples is that there is no requirement of any explicit rule for the task. In brief, learning in ANN can be seen as an automatic process of extracting rules from a data set [46]. Learning can be categorized in three types: supervised learning, unsupervised learning and back propagation. Each of these types is explained in this section.

3.5.1 Supervised Learning

In this type of learning, training patterns are composed of two parts: an input vector and an output vector coupled with input and output nodes respectively. The complete training cycle can be described in following steps:

I. An input vector is presented at the inputs along with expected or desired responses one for each node, at the output layer.

II. Then the actual response is calculated at the output layer and then the difference between actual calculated response and desired response is calculated.

III. The net difference between actual and desired response is used to adjust the weights in the neural net according to the prevailing learning rule or algorithm.

The term `supervised' was coined from the fact that the desired or expected responses on output nodes are provided by an external `teacher'. The teacher may be a set of training data or an observer who evaluates the performance of the network results. Common learning algorithms, which implement supervised learning, are back propagation algorithm, delta rule and perceptron rule. Back propagation algorithm is the most widely used algorithm with feedforward neural networks [42, 44, 45,46].

3.5.2 Unsupervised Learning

In this type, learning of artificial neural networks is conducted without any external influence or teacher. It is unique method in a sense that, ANN is given a set of inputs without any desired responses or outputs. The objective is to have the network itself begin to organize and use those inputs to modify its own neurons' weights or in other words, gets trained by its own. Sometimes unsupervised learning is also known as adaptation. Unsupervised learning is analogical to human brain learning; latter also learns by its own. However, success rate with unsupervised learning in ANN is low as compared to supervised learning [47].

3.5.3 Learning Laws

Learning laws are the mathematical algorithms used to update the connection weights. Hebb [48] invented the first learning law commonly known as Hebb's Rule. Most of the laws are some sorts of variations of Hebb's rule. In this section commonly used learning laws are briefly explained.

1 Hebb's Rule

Donald Hebb [48] was the first person to introduce the concept of learning rule in 1949 in his book "The Organization of Behavior". The basic rule he proposed is: If a neuron receives an input from another neuron and if both are highly active (mathematically have the same sign), the weight between the neurons should be strengthened. The rule is used to adjust the weights of a unit in an artificial neural network, so that the actual output of the network matches that of the expected or desired output. In Hebb's rule, the weights are initialized to 0 and weight adjustments (learning) are performed according to the equation 3.4.

$$W_i\,(\text{new}) = W_i\,(\text{old}) + X_i * T \qquad \{3.4\}$$

Where $\boldsymbol{W_i}$ is the weight associated with input $\boldsymbol{X_i}$ and $\boldsymbol{T}$ is the target output.

2 Perceptron Learning Rule

Rosenblatt [49] in 1958 introduced more powerful learning rule than Hebb rule known as perceptron learning rule. It is iterative learning rule and can give fair amount of convergence between desired response and calculated response. The rule is stated as follows:

- ❖ If the calculated output is the same as the expected (or desired) output, then do nothing.
- ❖ Else, if calculated output is 1 then decrement all active weights by 1. An active weight is the one, which leaves a unit whose output is 1.
- ❖ Else, if calculated output is , zero , increment all active weights by 1.

The perceptron training rule falls under supervised learning category since; implementer provides the target or expected outputs along with the inputs. Learning equation for the weights can be formulated as equation 3.5

$$W_i\,(\text{new}) = W_i\,(\text{old}) + \eta * X_i * T \qquad \{3.5\}$$

Where η is the learning rate ranges between $0 < \eta \leq 1$ and other symbols are explained in Hebb's rule.

3 Delta Rule

The delta rule was developed by Widrow and Hoff [50] to minimize the mean squared error. It is an enhancement of the Hebb's learning rule [48] that takes into account the error that is defined to be the difference between the desired response and the actual response. New weights are then assigned by addition of the error multiplied by the learning rate to the old weight. This learning rule efficiently brings the convergence between actual weight and ideal weight. For networks with linear activation functions and no hidden layers, a parabolic curved graph is observed between the error squared and the weights. Because of negative proportionality constant, the graph is concave shaped and has minimum value. The vertex of this paraboloid represents the point where the error is minimized. The weight vector corresponding to this point is then the ideal weight vector. The delta rule implements a gradient descent by moving the weight vector from the point on the surface of the paraboloid down toward the lowest point, the vertex. A detailed explanation of delta rule is given in [51].

4. Backpropagation Algorithm

Backpropogation learning algorithm is based on generalized delta rule. Basically, it is a gradient descent method, which minimizes the total squared error of the output computed by the neural net. Training is performed in cycles until a stopping criterion usually an acceptable error function is met. The negative of the gradient error function provides the direction in which the function decreases rapidly; hence the weights are adjusted accordingly. There are many available variations of this algorithm. A detailed discussion of backpropagation learning algorithm can be found in [51]. In our research we employed a feedforward backpropogation neural network as explained in Chapter 5.

3.6 Applications of Artificial Neural Networks

Despite of its invention in 1943, ANN started gaining popularity since the last two decades. Prominent areas where ANN is widely used are classification, pattern recognition, optimization, weather forecasting, stock market prediction and in many manufacturing applications. In this section we tried to explain few of the applications in brief as follows:

1 Classification

Classification is associated with the assignment of each object to a specific "class". It is of prime importance in a many areas such as image and speech recognition, social sciences etc. Vierrra and Ponz [52] applied the ANN for the automated classification of stellar spectra. Heckman et al. [53] presented an audio-video speech recognition system based on hybrid method of ANN and Hidden Markov Model (HMM) approach. Other practical classification tasks include recognizing printed or handwritten characters, classifying loan applications into credit-worthy and non-credit worthy groups [54].

2 Pattern Recognition

Pattern recognition is the field that deals with the operation and design of systems that recognize patterns in data. It encloses sub disciplines like discriminant analysis, feature extraction, error estimation, cluster analysis (statistical pattern recognition), grammatical inference and parsing (syntactical pattern recognition). Important application areas are image analysis, character recognition, speech analysis, man and machine diagnostics, person identification and industrial inspection. Do et al. [55] applied ANN for the identification of spiders by providing different images of spider species for ANN training. Chan and Sandler [56] applied neural network for the complete 2-D shape recognition system

3 Manufacturing Applications

ANN started gaining importance in the manufacturing applications only in last decade. As reviewed in Chapter 2, Section 2.2, typical areas include scheduling of machines, buffer optimization, system performance, cost estimation in packaging industries. Zheng et al. [21] applied the neural network approach to the early cost estimation of packaging products. They developed the back propagation neural network model for cost estimation based on design information only. They established the correlation between costs related features and the final cost of the product by training a back propagation network using historical data. Park et al. [25] studied the neural network approach along with heuristic rules to scheduling jobs on parallel machines. Sabuncuoglu and Touhami [8] did an experimental investigation of simulation metamodeling with neural networks and illustrated that simulation metamodels with neural networks can be effectively used to estimate the system performances. Bulgak et al. [7] studied the optimization of buffer sizes in assembly systems using ANN metamodel. They developed artificial neural network metamodel for simulation of an asynchronous assembly system and ANN metamodel together with simulated annealing (SA) was used to optimize the buffer sizes in the system.

CHAPTER 4
GENETIC ALGORITHMS: AN OVERVIEW

4.1 Introduction

Genetic Algorithms (GA) are global optimization techniques based on the evolutionary computation that avoid many of the fallacies exhibited by local search methods on difficult search spaces such as buffer allocation problem, machine sequencing problem, part routing problem, pattern recognition problem, etc. Genetic processes of biological evolutions are the basis of genetic algorithms. From biological perspective, it is concluded that an organism's structure and its ability to survive in its environment ("fitness") are determined by its DNA. An offspring, which is combination of both parents DNA, inherits traits from both parents and other traits that the parents may not have, due to recombination. These traits may increase an offspring's fitness, yielding a higher probability of surviving more frequently and passing the traits on to the next generation. Over time, the average fitness of the population improves [30].

According to the principles of natural selection and Darwin's theory of "Survival of the Fittest", natural population evolves over generations. In nature, individuals with the highest survival rate, have the comparatively large number of offspring. Highly adopted or fit individuals spread the genes to an increasing number of individuals in successive generations. The strong characteristics from different ancestors may sometime produce super fit offspring in future generation. The fitness of this offspring will be greater than that of either parent. Holland [57] was the first scientist who simulated the process of natural adoption on computer and provided the basis for application of GA to optimization area. Further, Goldberg [58] presented a number of applications of GAs in search, optimization and machine learning problems. In general, GA is a powerful technique for combinatorial optimization type of problems. GA does not find the exact optimal solutions, but it does find near optimal solutions with reasonable computational requirements [1, 4, 11]. GA starts with the selection of chromosomes from a given set of chromosomes in the population with subsequent application of crossover and mutation operators to these selected chromosomes.

4.2 Elements of Genetic Algorithms

All GAs have following elements in common: populations of chromosomes, selection based on fitness, crossover to produce new offspring and random mutation of new offspring. In GA terms, chromosome in population is represented as a string of genes. Each gene represents a variable from the system of interest. These genes can either be binary coded (0 or 1) for example 1100110 or coded as integers or real numbers for example 234567, depending on the type of application. Once the suitable representation scheme is selected, the next task is to generate initial population. In GA, generally the initial population is randomly generated. From many studies [5, 30, 31, 32], we can conclude that good GA should have population size between 30 and 100. Then selection crossover and mutation operators are applied to initial population in order to find the optimal solution for a given problem. Each of these operators is described in detail in section 4.3.

4.3 GA Operators

GA operators are used to alter the genetic composition of individuals or chromosomes. Genetic algorithms guarantees that population or chromosomes have a great chance to be evolved to the optimal solution [34]. GA starts with the selection of chromosomes from a given set of chromosomes in the population with subsequent application of crossover and mutation operators to these selected chromosomes. Crossover has higher probability while mutation has very low probability. Each of these operators is described in detail as follows:

Selection: - It involves with the selection of chromosomes from a given set of chromosomes in the population for reproduction. Chromosomes with higher fitness function have higher selection probability. Fitness proportionate selection is the most common method in selection. In this type of selection method, the number of times an individual is expected to be selected for reproduction is the ratio of its fitness to the average fitness of the population. "**Roulette Wheel Sampling**" is the simplest method of implementing fitness proportion selection [12]. The concept of roulette wheel sampling is giving each individual a sector of a circular roulette wheel equal in area based on the individual's fitness. The roulette wheel is then spun and the ball comes to rest on one wedge shaped sector, and the corresponding individual is selected. Selecting N chromosomes from population is equivalent to play N games on roulette wheel as each individual is drawn independently. However, other selection methods apart from roulette wheel selection are top mate selection, tournament selection and random selection.

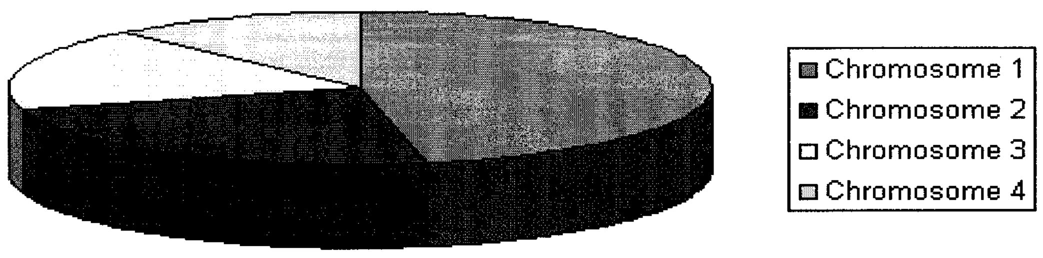


Figure 4.1 Roulette Wheel Selection

Top Mate Selection: In this type of selection, the first chromosome is selected based on fitness function while the second chromosome is selected randomly from the population.

Random Selection: In this type of selection, both chromosomes are randomly selected irrespective of the fitness function. This is the simplest form of selection.

Crossover: - This operator randomly chooses a locus or point and exchanges the subsequences before and after that locus or point between two chromosomes to create two offspring. For example, consider two parent chromosomes A and B with string 1100110 and 1001000 respectively. Suppose **single point crossover** takes place after 4'Th locus in a string of both chromosomes as shown in figure 4.1. The part of the strings of the parents A and B after the crossover point will be interchanged. Therefore, offspring A` and B` after crossover will have strings 1100000 and 1001110 respectively.

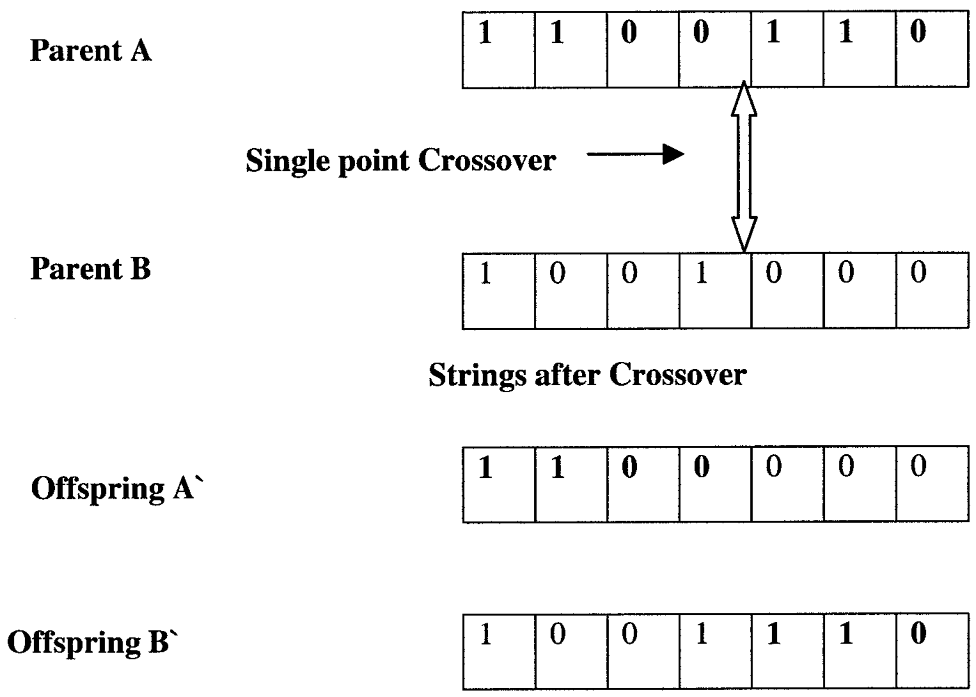


Figure 4.2 Single Point Crossover

In Multipoint crossover, the crossover takes place at two random locations. In this type, the part of the strings between two crossover points is interchanged as shown in figure 4.2.

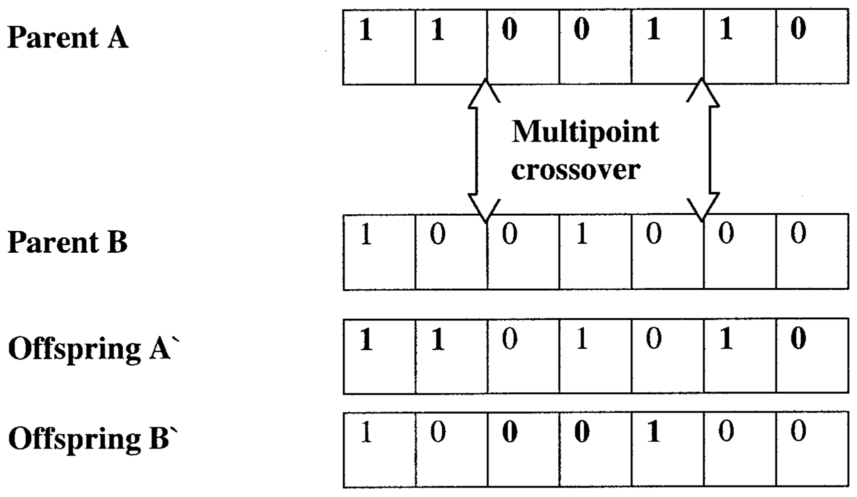


Figure 4.3 Multipoint Crossover

If no crossover takes place then offspring produced are the exact copies of their parents [12]. The crossover can be single point or multipoint crossover depending upon the type of GA. Simple GA has single point crossover

Mutation: - Mutation is one of the important genetic operators that randomly flip one or more bits in a chromosome for example in case of binary coding **1's** to **0's** and **0's** to **1's**. The objective of the mutation operator is to check the genetic population from converging to a local minimum and to introduce to the population new possible solutions. The Mutation takes place only after crossover. Mutation operation randomly changes the offspring resulted from crossover. The mutation is carried out according to the mutation probability Pm that is very low. Mutation can occur at any bit position in a string [12]. For example, consider an offspring A with chromosome structure 1101010 undergoes a mutation operator with probability P_m . Suppose the mutation occurred at 4'th bit position, then the gene 1, at 4'th bit position will change to 0 and the new offspring A\` will be 1100010 as shown in figure 4.3.

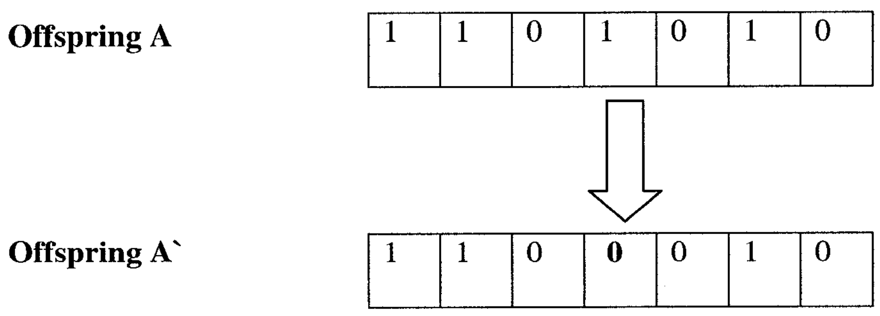


Figure 4.4 Mutation

4.4 Simple Genetic Algorithm

Genetic algorithms are inspired by Darwin's theory of evolution. Solution to a problem solved by genetic algorithms uses an evolutionary process (it is evolved). Algorithm begins with a **set of solutions** (represented by **chromosomes**) called **population**. Solutions from one population are taken and used to form a new population. This is motivated by an expectation, that the new population will be better than the old one. Solutions, which are then selected to form new solutions **(offspring)** are selected according to their fitness - the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Basic Genetic Algorithm

STEP I: Generate random population of *n* chromosomes (suitable solutions for the problem)

STEP II: Evaluate the fitness *f(x)* of each chromosome *x* in the population

STEP III: Create a new population by repeating following steps until the new population is complete

[Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

[Crossover] With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.

[Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).

1. **[Accepting]** Place new offspring in the new population
2. **[Replace]** Use new generated population for a further run of the algorithm
3. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
4. **[Loop]** Go to step **2**

For combinatorial optimization problems, GA works as follows: a population of solutions coded as strings of fixed length are maintained during the search. For each iteration, a new population P_{t+1} is created by retaining old solutions and generating new solutions from previous population P_t by applying three operators selection, crossover and mutation [5].

4.5 Termination Criterion

GA has a general behavior that, the solution of GA keeps improving over generations. But after a certain number of generations, convergence can occur such that, a fair amount of agreement between maximum fitness and average fitness can be observed [5]. Despite of the convergence behavior, GA has no convergence guarantee in arbitory problems [31, 32]. Detailed explanation of stopping criterion can be found in Diwan's work [59]. In our application we decided to use stopping criterion, which depend upon the size of the production system. For example in our study we decided to stop after 15 generations for a

7 stations assembly line. However if no fair agreement is observed at 15th generation, our GA search continues till convergence a criterion is observed.

4.6 Recent Applications of Genetic Algorithms

Genetic Algorithms became very popular in combinatorial optimization applications since the last decade. Variations in basic GA can be efficiently employed to solve the difficult problems such as NP Hard problems, Machine learning, pattern recognition and also for evolving simple programs. GAs are also being used in composing music [60]. Typical GA applications are described in brief as follows:

1) **Optimization:** GAs have been used in solving wide range of optimization tasks such as numerical optimization combinatorial optimization problems. As reviewed in Chapter 2 Section 2.3, typical combinatory optimization problem includes job shop scheduling, buffer optimization, and circuit board layout. Bulgak et al. [4] applied the schema theorem for solving buffer allocation problem in asynchronous assembly systems. Yokoyama and Lewis [38] applied the genetic algorithms for optimization of dynamic production cycling. Loh et al. [33] used genetic algorithm for sequential part assignment for printed circuit board assembly. Cochran et al. [36] developed a multi-population genetic algorithm to solve the multi- objective scheduling problems for parallel machines.

2) **Machine Learning**: This is one of the important areas where GAs have been widely used. Typical application includes classification and prediction tasks, such as prediction

of weather or protein structure [12]. Chen et al. [] applied genetic algorithms for information retrieval. Genetic algorithms can be effectively used to architecture and weights of the neural networks.

3) **Automatic Programming:** GAs have been used to evolve computer programs for specific tasks and to design other computational structures such as cellular automata and sorting networks [12].

4) **Economics:** GAs have been used to model processes of innovation, the development of bidding strategies and the emergence of economic markets [12].

5) **Social Systems:** GAs have been used to study evolutionary aspects of social systems, such as the evolution of social behavior in insect colonies, and more generally, the evolution of cooperation and communication in multi-agent systems [12].

Once, the basic GA algorithm implemented, we just have to write a new chromosome to solve another problem. With the same encoding, we simply change the fitness function and apply GA to other problem. However, for some problems, choosing and implementation of encoding and fitness function can be difficult.

GA is such a powerful heuristic search method that it can be applied to all of the areas. Some of the applications of GA are short listed as follows:

- Nonlinear dynamical systems - predicting, data analysis
- Designing neural networks, both architecture and weights

- Robot trajectory
- Traveling sales man problem and sequence scheduling
- Evolving LISP programs (genetic programming)
- Strategy planning
- Finding shape of protein molecules
- Functions for creating images

CHAPTER 5

DESIGN OPTIMIZATION OF FORK AND JOIN OPEN ASSEMBLY SYSTEM

5.1 Introduction

Production/assembly systems nowadays tend to be more expensive and complex because of the use of sophisticated equipment and flexible nature of the systems. Hence early design phase of such systems is highly significant. Decisions regarding product family, processes, machines, flow of the material etc are made in this phase. Hence, it is imperative to obtain feedback on the impact of such decisions on the system performance. However, complexity of the whole system makes initial judgment a dangerous design tool. Hence a model of a system is required to evaluate the performance of the alternative system configurations before actual implementation [59, 60]

In this chapter, we have presented two different ways of design optimization of fork and join open assembly systems. One way is the simulation optimization method in which, GA of Holland [57] works as an optimizer in search of an optimal buffer allocation for maximum production rate. The second method is simulation metamodeling, in which an ANN model was developed for simulation model of fork and join open assembly systems. Then GA was combined with ANN model to find the optimal buffer allocation for maximum production rate.

5.2 Design Optimization of Fork and Join Open Assembly System via Genetic Algorithm

5.2.1 Introduction

Fork and join assembly systems are increasingly being used in modern manufacturing systems. Due to the complexities in their configurations, designing such systems for an optimal performance may pose a number of challenges. This section presents a design optimization approach in fork and join open assembly systems. Optimal buffer allocations to accommodate the work-in-process inventories in such systems are optimized in an attempt to maximize the overall system production rate.

An assembly system consists of the series of workstations performing specified set of tasks repeatedly on consecutive product units moving along the line at constant speed. Each workstation takes the same amount of time to perform an operation on each unit. This operation time is called as workstation cycle time and reciprocal of workstation cycle time is called as production rate [32]. Depending on the type of the assembly system, these workstations are either manually operated or computer controlled. Also, workstations are connected to each other with some transfer mechanism such as conveyors or transfer chains or automated guided vehicles. Assembly systems are classified as close loop or open loop assembly system, and as synchronous or asynchronous assembly systems. Detail discussion of assembly system can be found in Chapter1, Section 1.3. In AASs, the transfer of parts between stations is handled by either transfer chain or conveyors or automated guided vehicles (AGVs). Assemblies are often transported on work carriers or pallets with fixtures that hold assemblies. The fixed number of pallets often circulates in the AASs as per system design requirements [2, 3, 4].

Asynchronous assembly systems are high-speed production lines. Failure of a single station may cause the complete stoppage of the entire assembly system. Apart from these regular failures, AASs are subject to station jams. Although, a station jam is a rare event, it may affect the overall system performance. Blocking and Starvation are the two important consequences associated with Jams. Suppose a jam occurs at an assembly station, station will stop processing assemblies for certain time until the operator clears the jam. During this temporary stoppage of the line, buffer space between the station experiencing jam and one or more upstream stations may get filled. The upstream station is then either unable to release the part in buffer space or the upstream station is blocked from releasing the finished assembly to be processed in next stations. Similarly because of the limited numbers of pallets, the jamming of station may lead to the starvation of downstream stations [2, 3].

5.2.2 Problem Statement

Because of the involvement of large system parameters and variables, designing an assembly system is not an easy task. A comprehensive design process involves a thorough study of the assembly system to determine the required or optimal system parameters and variables under the available technology constraints and required product quality standards with an objective that system would operate at a desired production rate to meet the targeted demand [5]. As explained in Chapter 1, Section 1.5, the design of an assembly system can be categorized in to resource requirement problem, resource layout problem, material flow problem and buffer capacity problem. A comprehensive design process includes optimization of all of the problems. However, the resource requirement problem, resource layout problem and material flow problem are out of the scope of this research. The objective of this study is the optimization of inter-stage buffers simultaneously considering other relevent design parameters such as jam rates, jam clear times, total number of pallets for fork and join open assembly system. Hence from now onwards, the term design optimization refers to the aforementioned context rather then a comprehensive design optimization.

The assembly system (Figure 5.2.1) under consideration is a fork and join open assembly system and is a modified version of the assembly system described by Hemchandra and Eedupuganti [8]. The topology of the system is described as follows:

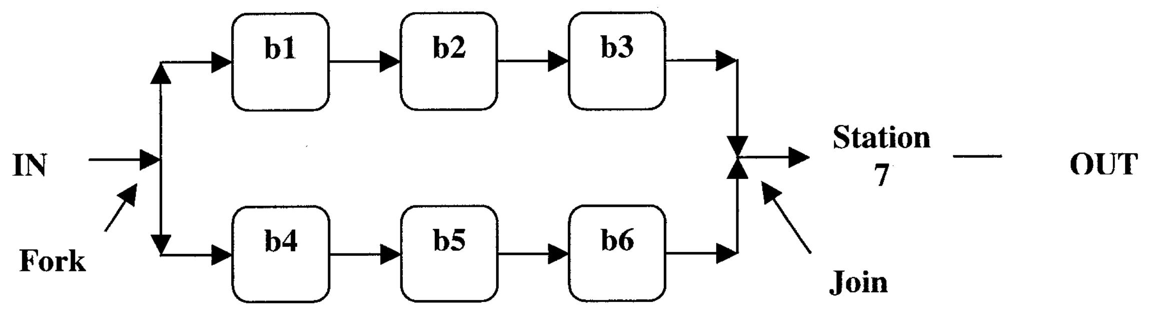


Figure 5.2.1 Topology of 7 Stations Fork and Join Open Assembly System

In this system, b1, b2, b3, b4, b5 and b6 represents the inter-storage buffer capacities between assembly stations 1 - 2, 2 - 3, 3 - 7, 4 - 5, 5 - 6 and 6 - 7 respectively. All stations are arranged in tandem according to the order of the assembly. Parts entering the system are sent either to station 1 or to station 4 depending on the process requirements. In brief, the system is divided into two subsystems, the one with stations 1, 2, 3 and the other with stations 4, 5, 6. At station 7, parts from these two subsystems are assembled and are then sent either to the next shop floor or to the warehouse. Also, we have assumed that there are always enough parts entering the system to accomodate the system throughput. Here, the system that we have considered is an automatic asynchronous type assembly system. The distance between any two adjacent stations (connected by transfer chain or conveyor) and the pallet dimensions determine the numbers of pallets than can be accommodated in between these adjacent stations. The maximum amount for this work in process (WIP) inventories between each pair of stations in the system (b1, b2, b3...in fig 1.2) constitutes the buffer sizes (capacities).

Selection of appropriate buffer sizes in an automated manufacturing system is a complex task, subject to arbitrary fluctuations in production rate and transportation delays that are part of material handling system. Large buffer sizes lead to excessive parts transfer delays and larger WIP inventories. Small buffer sizes leads to blocking of workstations in case of small processing delays or station jam. The production rate can be significantly affected by varying the inter stage buffer sizes. Hence its imperative that these inter stage buffer sizes be optimized in order to attain high system performance.

Hence, the buffer sizes between the pairs of stations constitute the vector of decision variables and the objective is to maximize the production rate. For a fixed number of pallets in the system, there is always an optimal buffer configuration capable of reducing the blocking and starvation effects to yield a maximum possible production rate [3, 4]. Fundamentally the problem is a stochastic, nonlinear, combinatorial optimization problem with discrete decision variables. Let us consider Y is the production rate, which is function of buffer sizes, and random variable v, which represents randomness of staions jam occurance and jam clear time. Hence our goal is to optimize the bufer sizes between stations while considering the effect of jam occurances to maximize Y.

The problem can be formulated as:

$$\textbf{Max: Production Rate} = \mathbf{Y}$$

$$\textbf{Where } \mathbf{Y = F\,(x);\ x} \textbf{ in } \mathbf{X}$$

$$\mathbf{F\,(x) = E_{v}\,f\,(x, v)}$$

Where, x is the vector of decision variable for buffer sizes b1, b2, b3 ......b6.

X is a set of constraints, (i.e. the upper bound constraint and the lower bound constraint for each buffer size).

5.2.3 Methodology

Since, the problem described in Section 5.2.2 is a stochastic, nonlinear combinatorial optimization type of problem with discrete decision variables; it is extremely difficult to find any closed-form expression to determine the expected value of the production rate. The production rate is estimated by taking the ratio of the number of finished assemblies at the last station to the total number of semi-finished assemblies in the system. Discrete event simulation is used to estimate the expected value of the production rate. The genetic algorithm of Holland works as an optimizer in search for an optimal buffer allocation [4]. Assembly system simulation and genetic algorithm are both implemented through computer programs written in Microsoft Visual Basic 6.0. Simulation model is developed in ARENA 5.0. As shown in figure 5.2.2, output from the simulation model is given to the optimizer (GA) to provide feedback on the progress of the search for an optimal buffer allocation. This feedback in turn acts as an input to the simulation model for further improvement [10]. The flowchart of the methodology is given in figure 5.2.3

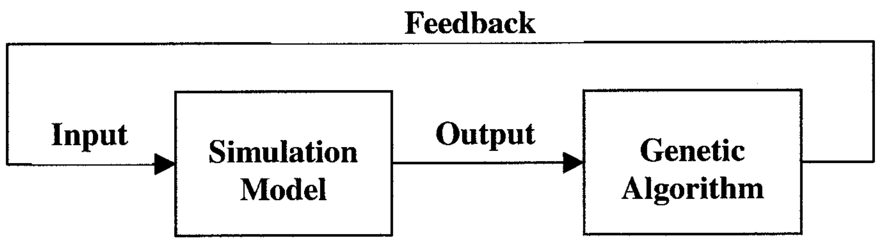


Figure 5.2.2 Flow of Methodology

5.2.4 Implementation

Fork and join open assembly system with different workstations and topology shown in figure 5.2.1 has been studied. Discrete event simulation and GA are applied to find production rate and optimal/near optimal buffer allocation for maximum production rate respectively. The important issues and parameters for assembly system and GAs are explained as follows.

Assembly Systems Issues

Workstation cycle time, station jam rates, jam clear time and number of pallets are the most important issues for the simulation of any production system. After going through various texts, research papers and our past experience with AASs, we decided upon following values of these issues.

Workstation Cycle Time: The time required to perform allotted task/work under normal conditions i.e. without any station jam. This is deterministic component of the workstation service time [3 ,4, 5, 59] and is considered as 5 time units.

Jam Rate: As explained in Chapter 1, Section 1.3, stations are subjected to jam due to various reasons. Theses are expressed in percentage for a particular station. For our system we decide to keep jam rate as 5% for stations 1, 5 and 6.

Jam Clear Time: This is expressed in time units required to clear a jam. In many studies [3, 4, 5] and consultations with design engineers, we understand that it is reasonable to assume an expected jam clear time approximately four times longer that the station cycle time. Also, geometric random variable could successfully model the jam clear duration's [5].

Number of Pallets: We decided to fixed, total number of pallets always circulating in the system. Since AAS are designed such that, it can accommodate 3 to 4 pallets between each pair of assembly stations. Also, it is important to note that considering the number of pallets, as a decision variable would yield an unstable optimization problem as in that both the number of pallets and buffer sizes would grow astronomically to ameliorate the production rate [5]. Hence, we decided to keep the total number of pallets as 30, always circulating in the system. It is important that, there should be some upper bound and lower bound for each buffer. In our application we kept upper bound for any buffer as 15 and lower bound as 1.

Simulation Parameters: Based on previous studies on AAS's [2, 3, 4], we decided to simulate the system around 10000 time units with 10 independent replications. To remove initial transient effects, a warm up time of 500 units is used for each replication.

GA Issues

After reviewing the literature on the studies of GA's and consultations with other researchers in this field, we concluded that GA requires high crossover probability, low mutation probability and modest population size. Based on these suggestions, we implemented following parameters;

Crossover Probability (P_c) = 0.6,

Mutation Probability (P_m) = 0.033

Population Size (n) = 30

Other characteristics of GA are described as follows.

Initial Population: Initial Population consists of binary coded buffer sizes with the production rate as fitness function. Each buffer space is represented by binary string of 4 (0's and 1's), for example string 1111 represents 15. Here we used the same representation scheme described by Bulgak et al. [4]. The initial population is randomly created and GA operators: selection, crossover and mutation are applied to this initial population. Then chromosomes in the population were decoded and the production rate was estimated by discrete event simulation.

Convergence and Stopping Criterion: GA has general behavior that, the solution of GA keeps improving over generations and generations. But after a certain numbers of generations, convergence can occur such that, the fair amount of agreement between maximum fitness and average fitness can be observed [4]. The stopping criteria depend upon the size of the production system. For example, in our study a convergence between maximum fitness and average fitness is reached at 15'th generation. However if no fair agreement is observed at 15th generation, our GA search continues till convergence criterion is observed.

Table 5.2.1: Iteration information for 7 stations Fork and Join Open assembly system with jam rates 0%, 5%, 0%, 0%, 5% and 5% for stations 1 to 6 respectively. Total Number of Pallets = 30

Gen No.	Binary Coded String	Buffer						Max Fitness/ Production Rate	Average Fitness/ Production Rate
		b1	b2	b3	b4	b5	b6		
0	010010010011001001010111	4	9	3	2	5	7	0.4643	0.4520
1	010010010011001001010111	4	9	3	2	5	7	0.4643	0.4557
2	010010010011001001010111	4	9	3	2	5	7	0.4643	0.4567
3	001010010001001010000111	2	9	1	2	8	7	0.4782	0.4645
4	001010100010000101110111	2	10	2	1	7	7	0.4874	0.4770
5	001010100010000101110111	2	10	2	1	7	7	0.4874	0.4778
6	001110010010001001100111	3	9	2	2	6	7	0.4892	0.4798
7	001110010010001001100111	3	9	2	2	6	7	0.4892	0.4820
8	001010000001001001111001	2	8	1	2	7	9	0.4894	0.4857
9	001010000001001010010111	2	8	1	2	9	7	0.4897	0.4798
10	001010000001001010010111	2	8	1	2	9	7	0.4897	0.4810
11	001010000010000101111001	2	8	2	1	7	9	0.4892	0.4889
12	001010010010000110001001	2	9	2	1	8	9	0.4896	0.4890
13	001010010010000110001001	2	9	2	1	8	9	0.4896	0.4892
14	001010010010000110010111	2	9	2	1	9	7	0.4898	0.4894
15	001010000010000110001000	2	9	2	1	9	7	0.4898	0.4896
Opt.	**001010010010000110010111**	**2**	**9**	**2**	**1**	**9**	**7**	**0.4898**	**0.4896**

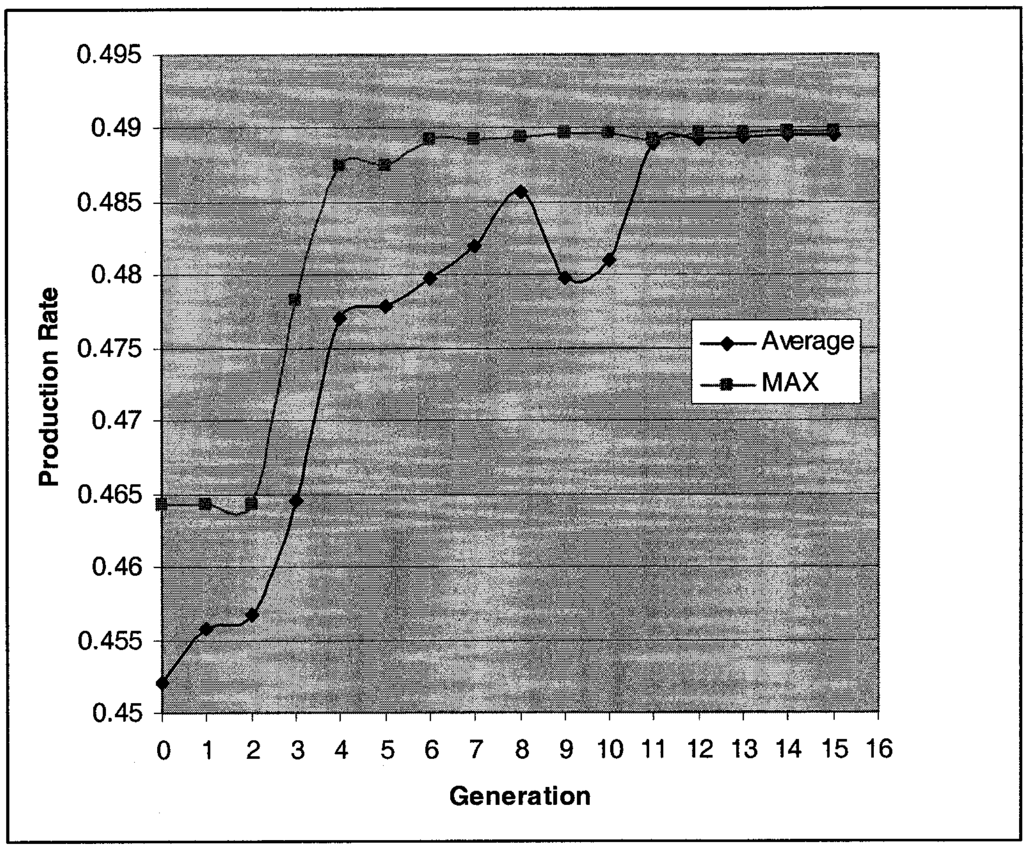


Figure 5.2.3 Convergence Behavior of Maximum and Average production rate of 7 stations Fork and Join Open Assembly System.

As shown in figure 5.2.3, a convergence between maximum production rate (0.4898) and average production rate (0.4895) is obtained at 14'Th generation. The corresponding optimum buffer sizes are 2, 9, 2, 1, 9 and 7. The idea behind tabulating average fitness is to exploit the fact that at a given point even if the maximum production rate reaches its peak, average fitness keeps improving as a law of genetic algorithms. Hence we can get greater values of average production rate. As shown in table 5.2.1, from generation 14-15 the maximum production rate is stagnant at 0.4898 while the average production rate increased from 0.4894 to 0.4896.

Table 5.2.2: Iteration information for 11 stations Fork and Join Open assembly system with jam rates 0%, 5%, 0%, 5%, 0%, 0%, 5%,0%, 5% , 5% for stations 1 to 10 respectively.

Generation	BUFFERS										MAX Prod Rate	AVG Prod Rate
	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10		
0	3	7	2	5	2	1	5	2	7	6	0.4187	0.3925
1	2	7	2	6	2	1	7	1	7	5	0.419	0.3974
2	2	7	2	6	2	1	7	1	7	5	0.4189	0.3986
3	1	6	3	7	2	3	5	2	6	5	0.4191	0.4015
4	1	6	4	5	3	3	5	2	6	5	0.4195	0.3996
5	1	7	3	7	1	1	7	2	6	5	0.4194	0.4067
6	1	5	2	7	1	3	7	3	6	5	0.4196	0.4084
7	1	5	2	6	2	2	7	3	7	5	0.4196	0.4093
8	2	5	2	6	2	4	4	3	7	5	0.4193	0.4098
9	2	8	1	5	2	2	4	3	8	5	0.4198	0.4067
10	2	8	1	4	2	2	7	2	7	5	0.4194	0.4093
11	2	6	1	8	2	1	5	2	8	5	0.4216	0.4155
12	1	6	2	7	2	1	5	3	8	5	0.4236	0. 4175
13	1	6	2	7	2	2	4	3	8	5	0.4236	0.4189
14	1	6	2	7	2	2	4	3	8	5	0.4236	0.4192
15	1	6	2	7	2	2	4	3	8	5	0.4236	0.4205
16	1	6	2	7	2	2	4	3	8	5	0.4236	0.4205
OPT	**1**	**6**	**2**	**7**	**2**	**2**	**4**	**3**	**8**	**5**	**0.4236**	**0.4205**

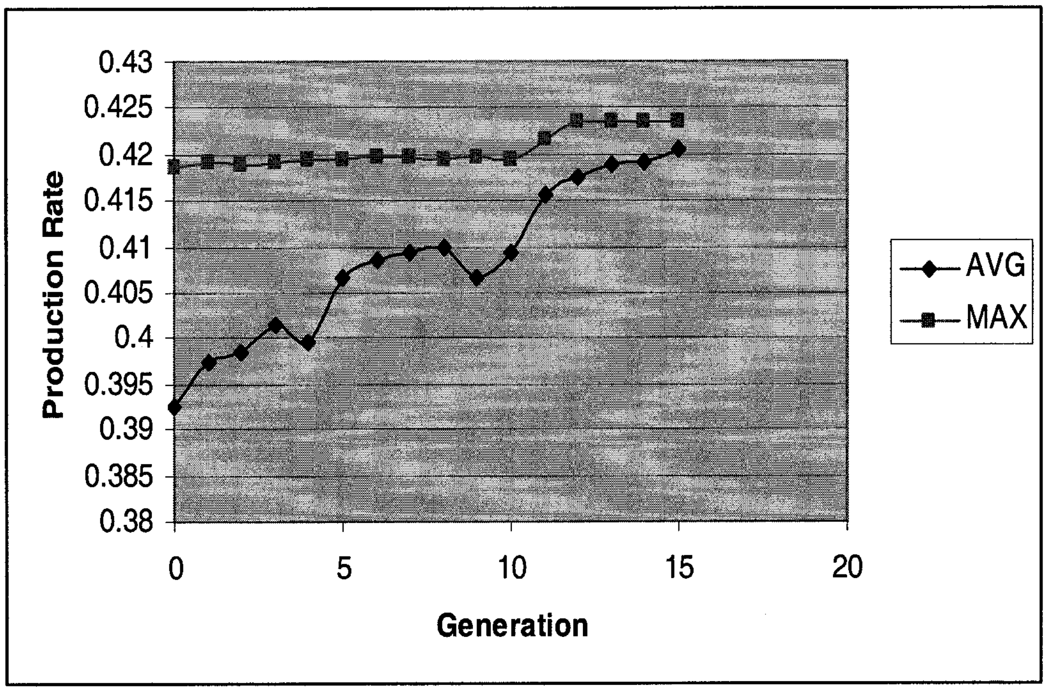


Figure 5.2.4 Convergence Behavior of Maximum and Average production rate of 11 stations Fork and Join Open Assembly System.

More examples of convergence behavior of maximum and average production rate for different configurations of fork and join open assembly systems are given in Appendix I.

5.3 Design Optimization via ANN and GA

5.3.1 Introduction

Simulation has been universally used by the manufacturing systems design engineers as a flexible tool in modeling and analysis of complex manufacturing systems. It reduces cost, time and risks associated with the implementations of new designs. However, simulation is extremely time consuming due to lengthy computational requirements for most of the real life problems. If a system has large parameter space, then simulation can be impractical for any parametric study of the system performance [7]. Systematic performance studies of most real world problems are beyond reach, even with supercomputers, unless substantial improvement in the speed of the performance evaluation process can be achieved. One approach to overcome this limitation is to develop a simpler model to explain the relationship between the inputs and outputs of the system. Metamodels are the models of simulation models and are increasingly being used in conjunction with the original simulation, in an attempt to improve the analysis and understanding of decision-making processes. A simulation metamodel is a simpler model of the real system. The simulation model is an abstraction of the real system in which a selected subset of inputs is considered. The effect of the excluded inputs is represented in the model in the form of the randomness to which the system is subject. Simulation is used to generate data sets, which in turn are used to build the metamodel. A simulation metamodel with neural networks is a neural network whose training is provided by a simulation model. In general, a metamodel takes a fewer number of inputs and is usually simpler than the simulation model. [8].

5.3.2 Metamodeling Concept

Although, the simulation model is simpler than the real world system, it is still a very intricate way of relating input to output. Sometimes, a simpler model may be used as an auxiliary to the simulation model in order to better understand the more complex model and to provide a framework for testing hypothesis about it. This auxiliary model sometimes referred to as a metamodel [61]. The definition of simulation metamodeling can be found in Kleijgen [62] and Chen et al. [27] and summarized as follows:

Let us consider X $(X = \{x_j \mid j = 1,2,..r\})$ and Y $(Y =\{ y_k \mid k = 1,2 ....n \})$ are the independent and dependent factors respectively. A factor may be a quantitative or may be qualitative. Then the response variable y_k can be defined as

$$y_k = f_1 (x_1, x_2 , ..., x_r) \qquad \{ 5.1\}$$

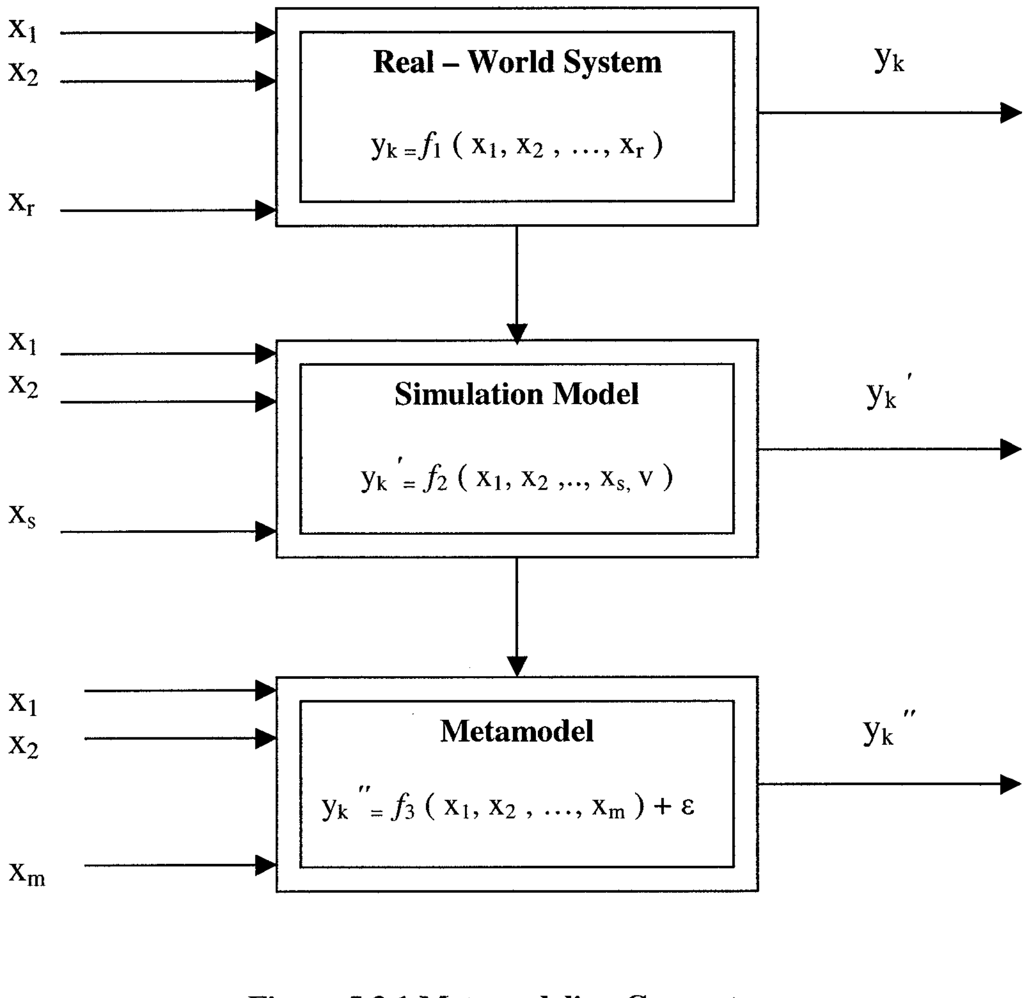
A simulation model is an abstraction of a system, in which we consider only a subset of the input factors $\{x_j \mid j = 1,2, ...s\}$. Generally, s is significantly smaller than the unknown value. Equation 5.2 defines the simulation response y_k' as function of f_2 and random numbers vector v , representing the effects of excluded outputs.

$$y_k' = f_2 (x_1, x_2 ,.., x_s, v) \qquad \{ 5.2\}$$

A metamodel is a model of simulation model and is further abstraction of the simulation input variables $\{x_j \mid j = 1,2, \ldots m;\ m \leq s\}$. Metamodel response as

$$y_k'' = f_3 (x_1, x_2 , \ldots, x_m) + \varepsilon \qquad \{ 5.3\}$$

Where ε denotes fitting error having an expected value of zero. Figure 5.3.1 shows the flow of three level abstraction of simulation metamodel.



x_1
x_2
x_r
Real – World System
$y_k = f_1 (x_1, x_2 , \ldots, x_r)$
y_k
x_1
x_2
x_s
Simulation Model
$y_k' = f_2 (x_1, x_2 ,.., x_s, v)$
y_k'
x_1
x_2
x_m
Metamodel
$y_k'' = f_3 (x_1, x_2 , \ldots, x_m) + \varepsilon$
y_k''

Figure 5.3.1 Metamodeling Concept

5.3.3. Application of Simulation Metamodeling to the Design Optimization of Fork and Join Open Assembly System.

A neural network is a proven tool in providing excellent response prediction in diverse application areas such as manufacturing, management, marketing accounting, finance etc. [63,64]. From recent survey [7, 8, 63, 64] neural networks shows promising results when compared to other techniques. In this study a design methodology incorporating Back Propagation (BP) based metamodeling and a GA based optimization technique is applied to the design optimization problem of 11 stations, Fork and Join open assembly system. The flow of the methodology is given in figure 5.3.2.

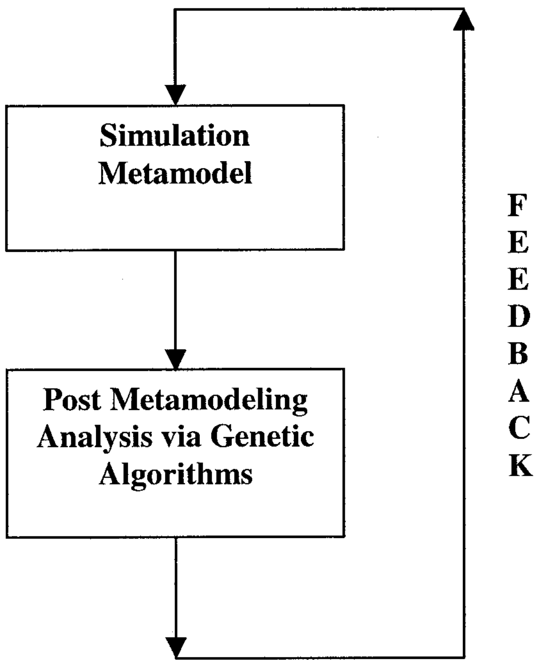


Figure 5.3.2 ANN Methodology

Figure 5.3.2 shows the hybrid methodology applied to design optimization problem of fork and join open assembly system. The discrete event simulation model was first developed for the problem under study. A set of training data and a set of testing data were generated through this simulation model. The training patterns (set of inputs with targeted outputs) were then fed into a BP network to learn the relationships between the input parameters and simulation responses of the interest [7, 27]. Once the neural network is trained, testing data is used to test the validity of ANN model. Then Genetic Algorithm in conjunction with the ANN model was applied to optimize the buffer sizes between the stations of fork and join open assembly system. After the creation of ANN metamodel, we don't need to run the simulation model in search of optimal buffer allocation; hence, the lengthy computational time will drastically be reduced. We, then compared our results obtained from ANN-GA method with the Simulation-GA method.

5.3.3.1 Development of Network Architecture

Performance of ANN model can be evaluated on several design parameters such as the number of layers in hidden layer, number hidden neurons in each hidden layer, the size of training set and training factors [7]. Figure 5.3.3 shows generalized ANN architecture with single hidden layer. Previous work [7, 64] in ANN showed that a single hidden layer is sufficient for ANN to approximate any complex nonlinear function as long as enough hidden neurons are used. It is imperative to decide number of hidden neurons in hidden layer at the beginning of building ANN. Second important factor in ANN building is size of the training data set. Since the size of the training data set influences the performance of ANN, it is essential to determine proper training size. The next important factor is selecting the learning algorithm. With our past experiences and from other studies [7, 8, 27, 64], we decided to use backpropgation-learning algorithm.

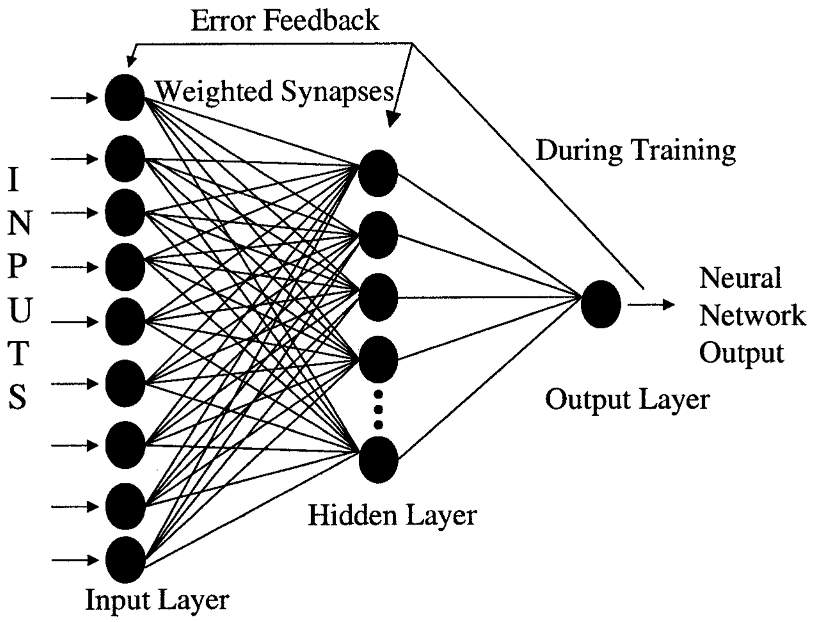


Figure 5.3.3 Artificial Neural Network

The value of learning coefficient (***l**$_c$*) significantly affects the effectiveness and convergence of backpropagation learning algorithm. The choice of ***l**$_c$* depends on the class of learning problem and the network architecture. A larger ***l**$_c$* will result in more rapid convergence, but overshooting is generally associated with larger ***l**$_c$*. Hence there are chances of lack of stabilization at any minimum. . Smaller ***l**$_c$* ensures true gradient descent, while the total number of learning steps to reach satisfactory results increases or in other words time required to reach convergence increases.

Till date, there is no method available to decide or set the optimum parameters for the building of ANN. Therefore we decided to do experimental investigation on the effect of number of hidden neurons in a single hidden layer, training data set and learning coefficient to find optimal or near optimal above mentioned parameters for ANN. Figures 5.3.4 to 5.3.7 shows the effects of learning coefficient on ANN performance.

Another important part in building ANN is the type of input to be given to train the ANN. Since the problem under consideration is finding an optimal set of buffers for maximum production rate with respect to total number of pallets always circulating in the system, stations subject to jams and their corresponding jam rates. From our previous studies [5, 7] we know that, the buffer sizes between the pairs of workstations and jam rate for each workstation significantly affect the production rate. Hence we decided to choose buffer sizes between the pairs of workstations and jam rate for each station as inputs to ANN model and output as production rate which is the function of inputs. Table 5.3.1 shows the sample input for ANN model. The training of ANN was carried out using backpropagation algorithm because of its powerful approximation capacity and its applicability to both binary and continuous inputs [7]. Neural Network toolbox of MATLAB 6.5 was used to develop and train ANN.

Table 5.3.1 Sample Training Data Set

Buffer's	**b1**	**b2**	**b3**	**b4**	**b5**	**b6**	**b7**	**b8**	**b9**	**b10**
	4	6	3	5	6	2	9	1	8	2

Stations	**WS1**	**WS2**	**WS3**	**WS4**	**WS5**	**WS6**	**WS7**	**WS8**	**WS9**	**WS10**
Jam Rates	0	5	0	5	0	0	5	0	5	0

After careful experimental investigation, we decided to keep 20 neurons in the input layer, 10 for buffer sizes and 10 for jam rates for various machines. Table 5.3.1 shows the sample input to the ANN. Also we deiced to keep a single hidden layer with 12 hidden neurons, and a single neuron in the output layer to map the objective function, which is the desired production rate. We used training data set of 150 different buffer configurations to train ANN.

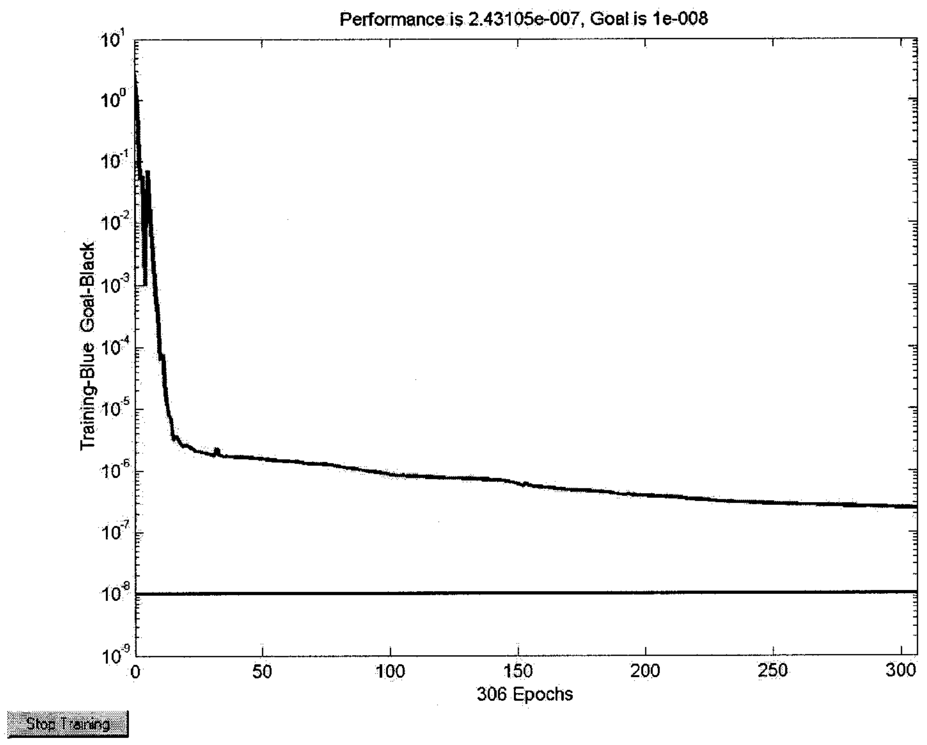


Figure 5.3.4 Performance of ANN for l_c = *0.4*

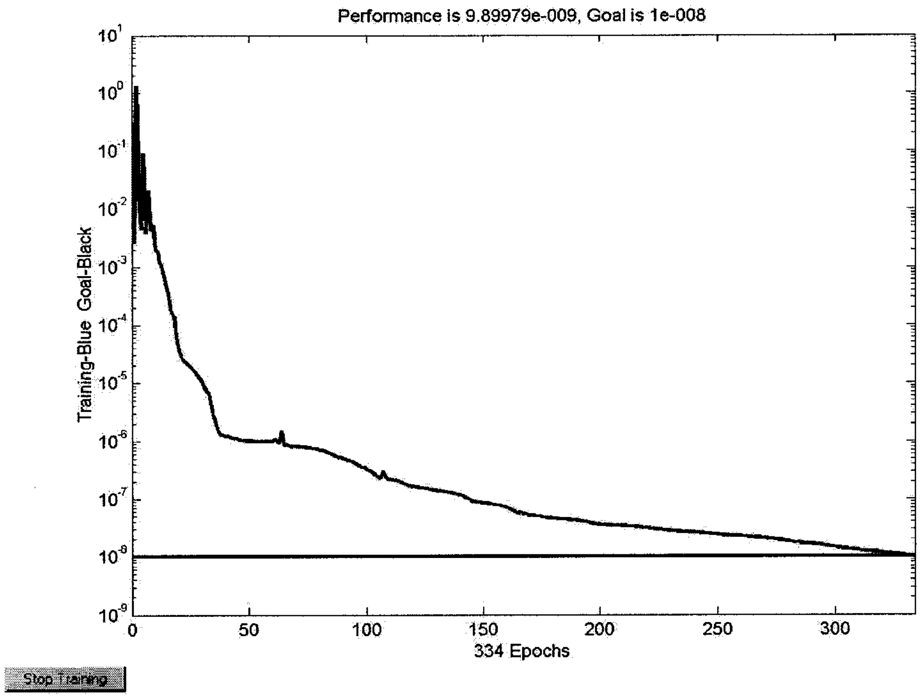


Figure 5.3.5 Performance of ANN for $l_c = 0.45$

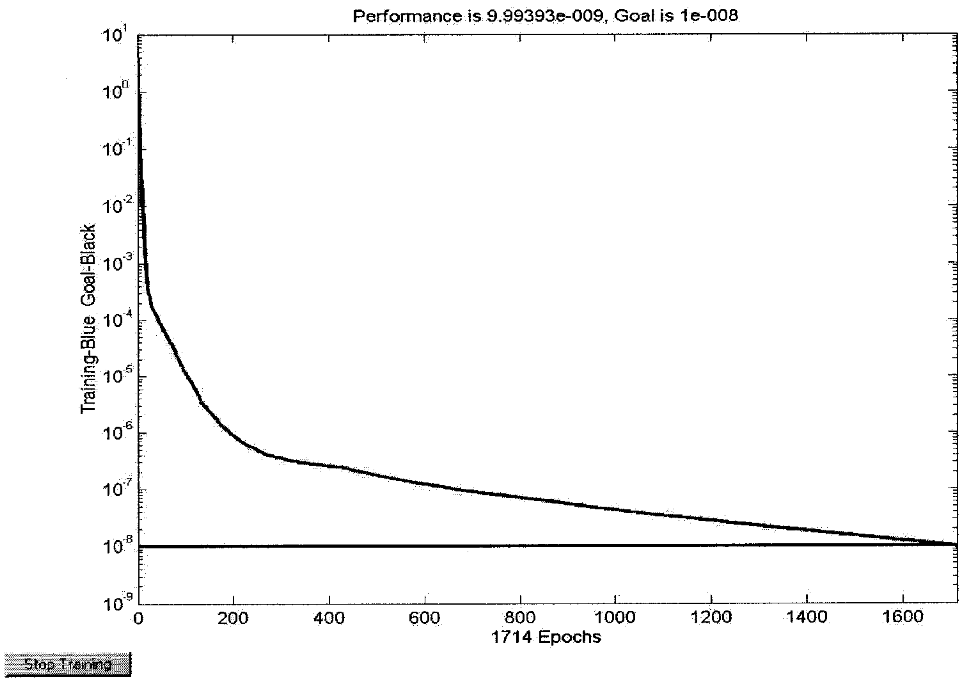


Figure 5.3.6 Performance of ANN for lc =0.5

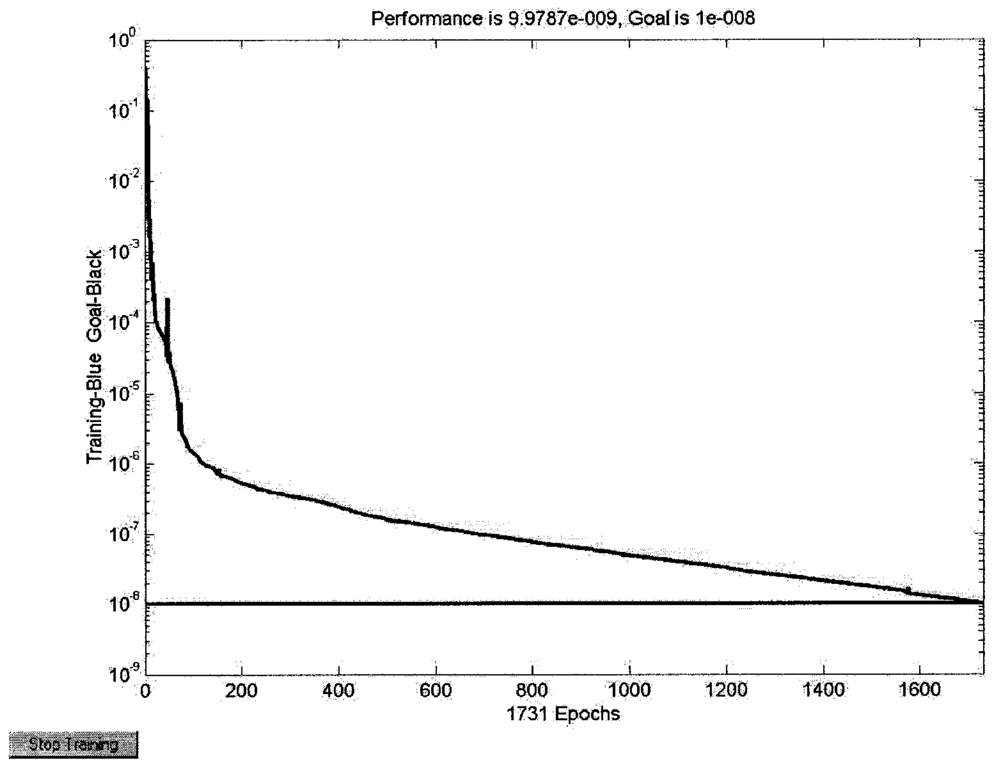


Figure 5.3.7 Performance of ANN for l_c=0.6

As seen in figure 5.3.4, the performance goal was not met for learning rate 0.4, but ANN reached its goal when we increase our learning rate to 0.45 (Figure 5.3.5). Then we further increase our learning rate to 0.5 and 0.6 and found out that there is no significant difference in error reduction. Hence we decided to keep our learning rate of 0.5.

5.3.3.2 Validation of ANN Model

Another important part of successful ANN model building is to validate ANN model. The main objective of validation is to ensure proper training of ANN model i.e to make sure that ANN model is neither over trained nor under trained and to evaluate the performance of ANN model after training. Since there is no well-specified or theoretical methodology available, we used a general approach for validating the ANN model. In this approach, a set of data, which was not used in the model building i.e unbiased data, was used to evaluate certain performance measures. The commonly used performance measures are: mean absolute error (MAE), root mean squared error (RMSE) and mean square error (MSE).

In our application, we used mean square error as a network performance measure and also we divided our data in to two parts: one is to train the network and another is to test the network. This approach is called as cross validation method. We then divided our test data in to two parts: one to ensure that our model is not over trained i.e to decide when we should stop training ANN model and second part is used after the training to estimate the error of the trained network [7]. The only disadvantage of this method is it requires large amount of data. Figure 5.3.8 shows the comparison of two test data sets and response from trained ANN model.

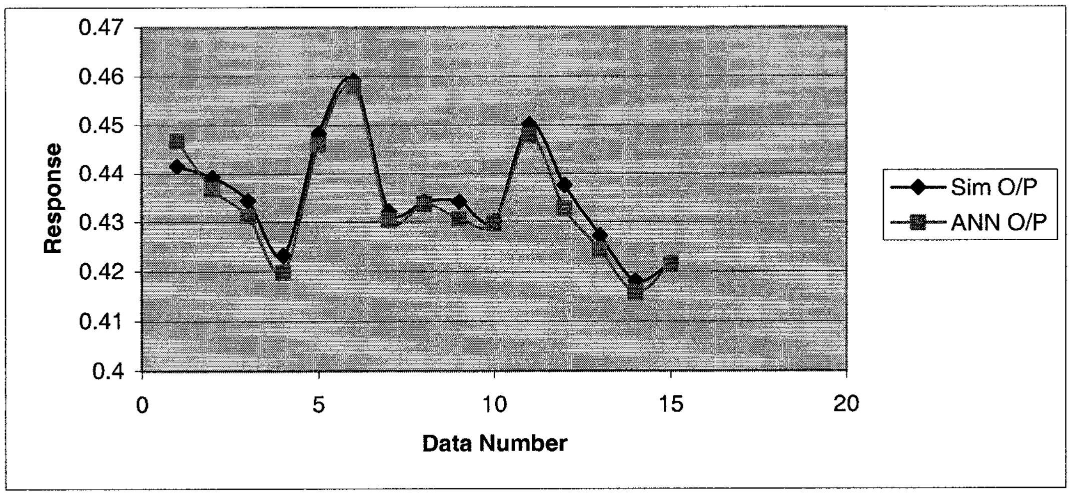


Figure 5.3.8 Comparison of ANN Output with Simulation Output

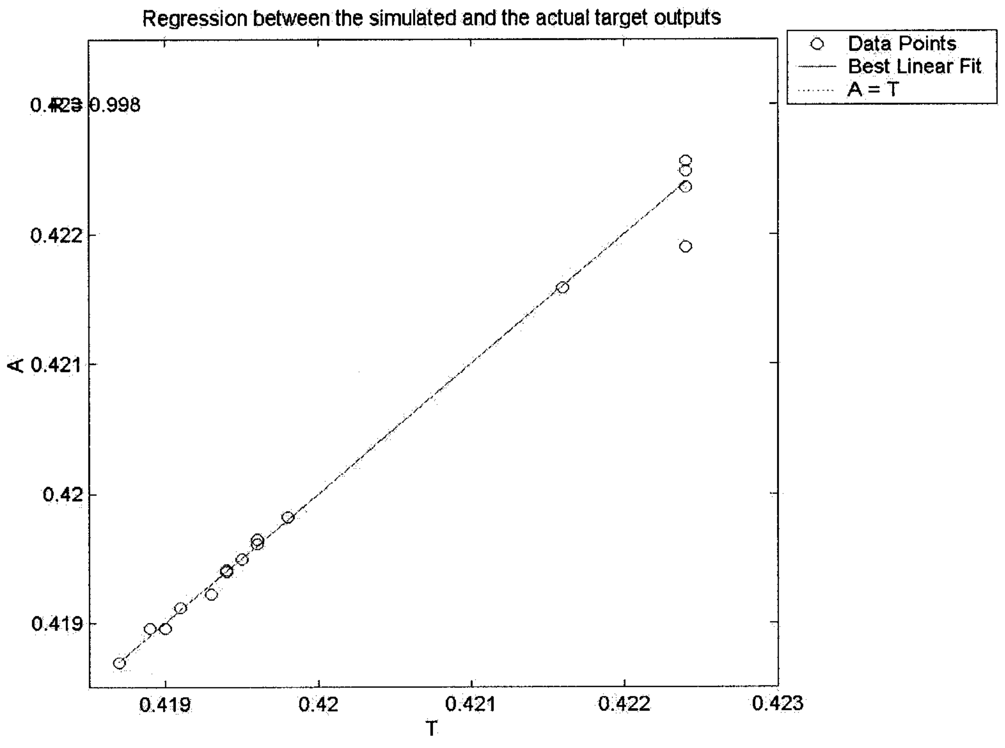


Figure 5.3.9 Post Regression Analyses

As per observation from graph, output from ANN is almost same as the output from simulation model, which is the indication that our ANN model is successfully built. Also as per regression analysis shown in figure 5.3.9, our actual and simulated outputs are best fitted. Now our model is ready to be coupled with any heuristic search algorithms to find the optimal solutions to our problem.

5.3.4 Post Metamodeling Analysis

Once the metamodel is developed, it can be coupled with variety of search methods to find optimal/near optimal solutions to the problems of interest. The advantages of metamodeling includes model simplification, enhanced exploration and interpretation of the model, generalization to other models of the same type, sensitivity analysis, optimization and better understanding of the studied system [27,66]. Also Kleijnen [62] identified four general goals for metamodels: understanding the problem entity, predicting the response value, aiding the verification and validation and performing the optimization (simulation-optimization). In this study, we developed an ANN metamodel for fork and join open assembly system to locate optimal/near optimal buffer allocation for maximum production rate. There are many simulation optimization methods available to solve complex problems. However most of them suffer with limitations such as trapping in local optima, high computation requirements [67]. Recently, more powerful heuristic search algorithms have been popular in many areas to over come limitations of other optimizing techniques. The most common heuristic search methods are simulated annealing, genetic algorithms, evolutionary programming and tabu search [7, 8, 27]. In this study we developed GA based optimization method to solve design optimization problem. Detail explanations of GA have been given in Chapter 4.

After preliminary study and going through various literatures [5, 7, 30, 31, 32, 34], we decided upon following parameters for GA.

Population Size: 50

Crossover Rate: 0.6

Mutation Rate: 0.03

Rest of the parameters, coding of chromosomes, and stopping criterion are same as explained in section 5.2. Table 5.3.2 shows the optimal buffer configuration obtained through ANN-GA method.

Table 5.3.2 Optimal Buffer Configuration by ANN-GA Approach

b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	Prod Rate
1	7	2	6	2	2	3	2	8	5	0.4261

5.4 Comparison of Simulation-GA with ANN-GA

In this study we first developed simulation model for fork and join open assembly system. Then GA was coupled with simulation model to find optimal buffer configuration for maximum production rate. We first studied 7 stations line and then same application domain was extended to 11 stations fork and join open assembly system. Our SGA method gives optimal solution in only 16 generations, but time required to attain optimal solution is very high. Hence we decided to look for an alternative approach, which can reduce computational time significantly, and we settled down for simulation metamodeling approach. Complete ANN model development process is given in Section 5.3. ANN-GA approach gives optimal solution in 24 generations but the computational time is much more less as compared to SGA approach. Also with ANN-SGA, we have more flexibility if we decided to change any parameter of interest. For example, if we decided to change the jam rate for any of the machines then we just have to enter the new jam rate for the machine .We do not have to run the simulation model again. Hence our lengthy computational time can be saved and quick decisions can be made regarding the design of assembly system. Figure 5.3.10 shows the comparison of iteration information for SGA and ANN-GA approach. We included maximum and average production rate for our analysis. As shown in figure 5.3.10, SGA approach gives optimal solution in just 16 generations, where as ANN-GA approach gives the optimal solution in 24 generations. However, despite of getting optimal solution in 24 generations, time required to reach the optimal solution in ANN-GA approach is much more less as compared to SGA approach. Hence quick decisions can be made to meet production requirements. More examples of convergence behavior of maximum and average production rate for different configurations of fork and join open assembly systems are given in Appendix II.

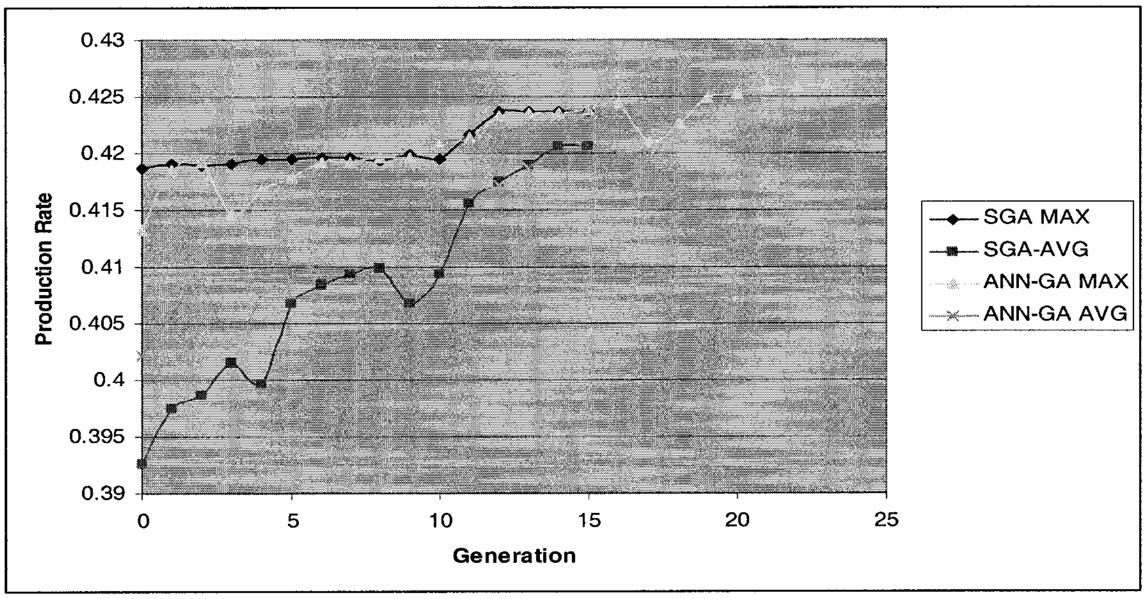


Figure 5.3.10 Comparison of ANN-GA and SGA Approaches

CHAPTER 6
CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This research presents two different methods for solving complex design optimization problem for Fork and Join open assembly systems. The first approach is coupling of a simulation model of the system under consideration with genetic algorithm to search optimal configurations of buffer sizes between each pair of work stations while considering different parameters such as jam rates, jam clear times. The results obtained indicate that genetic algorithms used in conjunction with simulation can effectively solve complex design problems such as the one studied in this research. However, if we want to do any parametric study and especially for large systems, simulation is extremely time consuming due to lengthy computational requirements.

As a result, we tested the second approach, i.e simulation metamodeling. We developed ANN network for the simulation model of fork and join open assembly system. However ANN network development is complicated task, but once the network is developed, it is relatively easy to change the parameters of the interest and do the analysis. Metamodels can also be coupled with variety of optimization algorithms. Simulation metamodeling proved to be an effective tool in overcoming fallacies of simulation models. Metamodels are much simpler, and more flexible as compared to simulation models. Results indicate that, ANN coupled GA can effectively solve the complex design optimization problem much faster than SGA approach but it requires large amount of sample data and expertise to built an ANN metamodel. We can conclude that ANN metamodeling proved to be an effective tool in manufacturing applications. However more investigation is needed in different manufacturing application areas.

The application of the simulation metamodeling approach to the design optimization of complex topologies of AASs such as incorporating inspection machines, rework centers and comprehensive comparisons of performances of different heuristic search algorithms constitutes our future research interests. Also one of the interesting topics to look into is the application of genetic algorithms for developing and training of artificial neural networks.

REFERENCES

1. W.M. Chow, 1990, “Assembly Line Design” Marcel Dekker Inc, New York.

2. Armin Scholl, 1995, “Balancing and sequencing of Assembly Lines”, Spriner –Verlag publications.

3. A.A.Bulgak and J.L.Sanders, 1991, “Modeling and design optimization of asynchronous flexible assembly systems with statistical process control and repair”, International Journal of flexible Manufacturing Systems, 3, 251-274.

4. A.A.Bulgak and J.L.Sanders, 1991, “Approximate analytical performance models for automatic assembly systems with statistical process control and automated inspection”, J. Manufacturing systems 10, 121-133.

5. A.A.Bulgak, P.D.Diwan and B.Inozu, 1995, “Buffer size optimization in Asynchronous Assembly Systems using Genetic Algorithms”, Computers Ind. Engng. Vol. 28 No 2,309-322.

6. George Chryssolouris, 1992, Chapter 5, “Manufacturing Systems Theory and Practice”, Springer-Verlag Publications, 221-311.

7. Akif A. Bulgak, Fulya Altiparmak and Berna Dengiz, 2002, “Optimization of buffer sizes in assembly systems using intelligent techniques” winter simulation conference

8. Ihsan Sabuncuoglu and Souheyl Touham, 2002, “Simulation metamodelling with neural networks: an experimental investigation” International Journal of Production Reseach, vol 40,No 11,2483-2505.

9. Alexandre Dolgui and Dimitry Ofitserov, 1997, “A stochastic method for discrete and continuous optimization in manufacturing systems” Journal of Intelligent Manufacturing, 8, 405-413.

10. V.Kouikoglou and Y.Phillis, 1997, “Continuous flow model for production networks with finite buffers, unreliable machines and multiple products”, International Journal of Production Res., Vol. 35, No. 2, 381-397.

11. Ayse Kavusturucu and Surendra M. Gupta, 1998, “ A methodology for analyzing finite buffer tandem manufacturing systems with N- policy” Computers and Ind Engng, Vol 34, No 4, 837-848.

12. H.Papadopoulos and M.Vidalis, 1999, “Optimal buffer allocation problem in short μ-balanced unreliable production lines”, Computers and Industrial Eng. 37, 691-710.

13. A. A. Bulgak, Y. Taracki and V. Verter, 1999, "Robust design of asynchronous flesible assembly systems" Internation Journal of Production Research, vol37 no 14. 3169- 3184

14. K. C. Jeong and Y. D. Kim, 2000, "Heuristics for selecting machines and buffer capacities in assembly systems" Computers and Industrial Eng, 38,341-360

15. Man-Soo Han and Dong-Jo Park, 2002, "Optimal buffer allocation of serial production lines with quality inspection machines" Computers and Industrial Eng, 42, 75-89.

16. C. H. Paik, H.G. Kim and H.S. Cho, 2002, "Performance analysis for closed loop production systems with unreliable machines and random processing times" Computers and Industrial Eng, 42,207-220.

17. Tom-David Graupner, H. Richter and W. Sihn , 2002, "Configuration, simulation and animation of manufacturing systems via internet" Proceedings of the winter simulation conference.

18. N.Hemchandra and S.K.Eedupuganti, 2003, "Performance analysis and buffer allocations in some open assembly systems" Computers & Operations Research 30, 695-704.

19. R. Inman, D.Blumenfeld, N. Huang and J.Li, 2003, "Designing production systems for quality: research opportunities from an automotive industry perspective", International Journal of Production Res., Vol. 42, No. 9, 1953-1971.

20. V.V. Emelyanov and S.I. Iassinovski, 1997, "An AI-based object oriented tool for discrete manufacturing systems simulation" Journal of Intelligent manufacturing 8, 49-58.

21. Y.F.Zhang and J. Y. H. Fuh, 1998, "A neural network approach for early cost estimation of packaging products" Computers and Industrial engng, vol 34, No 2, 433-450.

22. R. A. Kilmer, A.E. Smith, L.J.Shuman, 1998, " Computing confidence interval for stochiastic simulation using neural network metamodels" Computers and Industrial Engineering, special issue on artificial intelligence.

23. In Lee and Michael Shaw, 2000, "A neural network approach to real time shop flow sequencing" Computers and Industrial Engineering , 38, 125-147.

24. M. Haouani , M Ferney, N. Zerhouni and A. Elmoudni , 1995 , " Control of Manufacturing systems using neural networks" EUROSIM, 1163-1168.

25. Y. Park, S. Kim and Y.H.Lee, 2000, "Scheduling jobs on parallel machines applying neural network and heuristic rules" Computers and Industrial Engineering, 38, 189-202.

26. B. Dengiz, Akbay and S. Kunter, 2000, "Computer simulation of a PCB production line: metamodeling approach" International Journal of Production Economics, 195-205

27. M. C. Chen and T. Yang, 2002, "Design of manufacturing systems by a hybrid approach with neural network metamodelling and stochastic local search" International Journal of Production Research , 40, Vol 1 , 71-92.

28. K.Y.Jang, K. Yang and C kang, 2003, "Application of artificial neural network to identify non random variation pattern on the run chart in automotive assembly processes" International Journal of Production Research, Vol 41, 6, 1239-1254.

29. H. Ghaziri and I. H. Osman, 2003, "A neural network algorithm to traveling salesman problem with backhauls" Computers and Industrial Engineering, 44, 267-281.

30. Y. Carson and A. Maria, 1997, "Simulation Optimization: Methods and Applications", Proceedings of the 1997 winter simulation conference.

31. Diomidis D. Spinellis and C. T Papadopoulos, 2000, "Stochastic Algorithms for Buffer allocation in reliable production Lines", Mathematical Problems in Engineering, 5,441-458.

32. I. Sabuncuoglu, E. Erel and M. Tanyer, 2000, "Assembly line balancing using genetic algorithms" , Journal of Intelligent Manufacturing 11, 295-310.

33. T. S. Loh, S.T.S. Bukkapatnam, D Medeiros and H. Kwon, 2001, "A genetic algorithm for sequential part assignment for PCB assembly" Computers and Industrial Engineering, 40, 293-307.

34. G. Zhou, H .Min and M. Gen, 2002, " The balanced allocation customers to multiple distribution centers in the supply chain network: a genetic algorithm approach" Computers and Industrial Engineering, 43, 251-261.

35. C.T.Su and T. L.Chiang, 2003, " Optimizing the IC wire bonding process using a neural networks / genetic algorithms approach" Journal of Intelligent Manufacturing, 14, 229-238.

36. J K Cochran, S.M.Hrong and J. W. Fowler, 2003, "A multi-Population genetic algorithm to solve multi objective scheduling problems for parallel machines" Computers and Operations Research, 30, 1087-1102.

37. In. Chan Choi, Seong-In Kim and Hak Soo Kim, 2003, " A genetic algorithm with mixed region search for the asymmetric traveling salesman problem" Computers and Operations Research, 30 , 773-786.

38. Masao Yokayama and H. W. Lweis, 2003, "Optimization of stochastic dynamic production cycling problem by a genetic algorithm" Computers and Operations Research, 30, 1831-1849.

39.http://ieee.uow.edu.au/~daniel/software/libneural/BPN_tutorial/BPN_English/BPN_English/node5.html.

40. Dave Anderson and George McNeil, 1998, "Artificial Neural Networks Technology" technical report, data & analysis center for software, Rome- NY.

41. McCulloch, W. S. and W. Pitts, 1943, "A logical calculus of ideas immanent in nervous activity." Bulletin of Mathematical Biophysics, 5, 115-133.

42. K. Meherotra, C Mohan and S Ranka, 1997, "Elements of Artificial Neural Networks" MIT press publications.

43. Albrecht Schmidt, 1996, "A Modular Neural Network Architecture with Additional Generalization Abilities for High Dimensional Input Vectors." Masters thesis, Department of computing, Manchester Metropolitan University.

44. Dave Anderson and George McNeil, 1998, "Artificial Neural Networks Technology"

technical report, data & analysis center for software, Rome- NY.

45. J.A. Anderson, 1995, "Intorduction to Neural Netowrks" Cambridge MA: MIT press.

46. S. Haykin,1999, " Neural Networks: A Comprehensive Foundation " second edition,Englewood Cliffs , NJ:Prentice Hall.

47. Errki Oja,2002, " Unsupervised learning in Neural Computation" Journal of theorotical neural computing, vol 287 , Issue 1,187-207.

48. D. Hebb, 1949 , " The Organization of Behaviour" Wiley, N.Y

49. F. Rosenblatt, 1958, "The perceptron: A probabilistic model for information storage and organization in the brain," Psychological Review, vol. 65, 386—408.

50. S. Haykin, 1999, "Neural Networks. A Comprehensive Foundation", Second Edition, Prentice-Hall, Inc., New Jersey, 1999.

51. Mohammad Hassoun, 1995, "Fundamentals of Artificial Neural Netowrks" , MIT press publications, ISBN 0-262-08239-X.

52. E. Vierrra and J. Ponz, 1998, "Automated Spectral Classification using Neural Networks" ASP conference series, VOL 145 508-512.

53. M. Heckman, F. Berthommier and K. Kroschel, 2002, " A Hybrid ANN/HMM audio vedio sppech recognition system" Unpublished paper

54. http://www.neuroxl.com/finance_neural_network.htm

55. M.T. Do, J. Harp and K. Norris, 1999, " A test of pattern recognition system for identification of spiders" Bulletin of Entomological Research, 89, 217-224.

56. C.K. Chan and M.B. Sandler, 1992, "A complete shape recognition system using the Hough transform and neural network", 11th IAPR International Conference on Pattern Recognition Methodology and Systems, Vol II , 21-24.

57. J. Holland, 1973, "Genetic algorithms and the optimal allocations of trials, SIAM Journal of Computing, vol 2, no 2, pp. 88 - 105,

58 D.E.Goldberg, 1989, "Genetic Algorithms: In search, optimization and machine learning" Addison-Wesley publishing company.

59. P.D.Diwan, 1994, "Design Optimization and cost modeling of asynchronous automated assembly system using genetic algorithms" Master's thesis submitted at Concordia University, Montreal, Canada.

60. M. Towsey, A Brown, S. Wright and J. Diederich, 2001, "Towards melodic extension using genetic algorithms" Educational Technology and Society 4(2) 54-65.

61. Linda Weiser Friedman, 1996, "The Simulation Metamodel", kluwer academic publishers ISBN 0-7923-9648

62. J. Kleijnen, 1979, "Regression metamodels for generalizing simulation results", IEEE Transactions on Systems, Man and Cybernetics, 9(2), 93-96.

63. A. Vellido, P.J.G Lisoba and J Vaughan, 1999, "Neural networks in business: a survey of applications "Expert Systems with Applications, 17, 51-70.

64. H.C.Zhang and H. Huang, 1995, "Applications of neural networks in manufacturing: a state of art survey", International Journal of Production Research, Vol 33, 3, 705-728.

65. G. Cybenco, 1989, "Approximation by superposition of a sigmoid function" Mathematical Control Signals Systems, 2, 303-314.

66. L.W. Friedman and I. Pressman, 1988, "The metamodel in simulation analysis: can it be trusted?" Journal of Operational Research Society, 39(10), 939-948.

67. H. Pierreval. and J.L Paris, 2000, "Distributed Evolutionary Algorithms For Simulation Optimization", IEEE Transactions On Systems, Man And Cybernetics-Part A: Systems And Humans, 30(1), 15-24.

APPENDIX

- **Appendix I: Convergence behavior of Max and Avg production rate for different configurations of Fork and Join Open assembly systems via Simulation-GA approach.**

- **Appendix II: Convergence behavior of Max and Avg production rate for different configurations of Fork and Join Open assembly systems via Metamodeling-GA approach.**

Appendix I: Convergence behavior of Max and Avg production rate for different configurations of Fork and Join Open assembly systems via Simulation-GA approach.

Table 1A: Iteration information for 7 stations Fork and Join Open assembly system with jam rates 0%, 3%, 2%, 0%, 2% and 0% for stations 1 to 6 respectively. Total Number of Pallets = 30

Generation	Buffers						MAX Prod Rate	AVG Prod Rate
	b1	b2	b3	b4	b5	b6		
0	4	5	6	4	7	4	0.4657	0.4496
1	4	6	5	4	8	3	0.4656	0.451
2	4	6	5	5	7	3	0.4674	0.4533
3	4	4	7	6	6	3	0.4673	0.4542
4	4	4	7	6	6	3	0.4673	0.4551
5	4	4	7	6	6	3	0.4673	0.4559
6	3	4	7	6	7	3	0.4675	0.4571
7	3	4	6	5	8	4	0.4677	0.4568
8	3	4	6	4	8	5	0.4681	0.4574
9	3	4	6	4	8	5	0.4681	0.4576
10	3	4	6	4	8	5	0.4681	0.4575
11	3	4	7	4	8	4	0.4683	0.4578
12	3	4	7	4	8	4	0.4683	0.4579
13	3	4	7	4	8	4	0.4683	0.4579
14	3	4	7	4	8	4	0.4683	0.4579
OPT	**3**	**4**	**7**	**4**	**8**	**4**	**0.4683**	**0.4579**

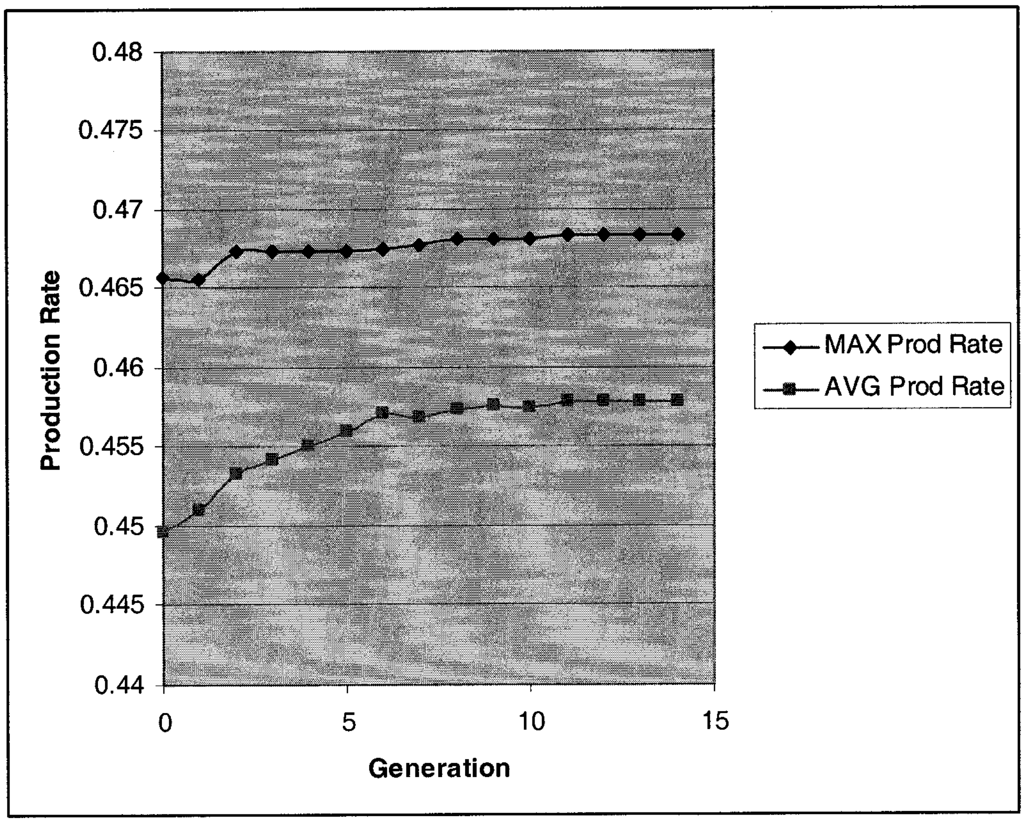


Figure 1A Convergence Behavior of Maximum and Average production rate (Table 1) of 7 stations Fork and Join Open Assembly System.

Table 2A: Iteration information for 7 stations Fork and Join Open assembly system with jam rates 0%, 2%, 3%, 0%, 3% and 3% for stations 1 to 6 respectively.
Total Number of Pallets = 40

Generation	Buffers						MAX Prod Rate	AVG Prod Rate
	b1	b2	b3	b4	b5	b6		
0	5	7	9	4	7	8	0.4887	0.477
1	5	8	10	3	7	7	0.489	0.4774
2	4	8	10	3	8	7	0.4889	0.478
3	5	7	8	4	9	7	0.4891	0.4786
4	4	7	9	3	9	8	0.4884	0.4782
5	5	7	9	3	9	8	0.4884	0.4782
6	5	6	9	3	9	8	0.4879	0.4776
7	5	7	8	3	10	7	0.4886	0.4785
8	5	7	8	3	8	9	0.4888	0.4791
9	5	9	6	3	9	8	0.4897	0.4795
10	5	9	6	3	9	8	0.4897	0.4798
11	5	9	6	3	9	8	0.4897	0.4801
12	4	10	6	3	9	8	0.4902	0.4806
13	4	10	6	3	9	8	0.4902	0.4808
14	4	10	6	3	9	8	0.4902	0.4808
OPT	**4**	**10**	**6**	**3**	**9**	**8**	**0.4902**	**0.4808**

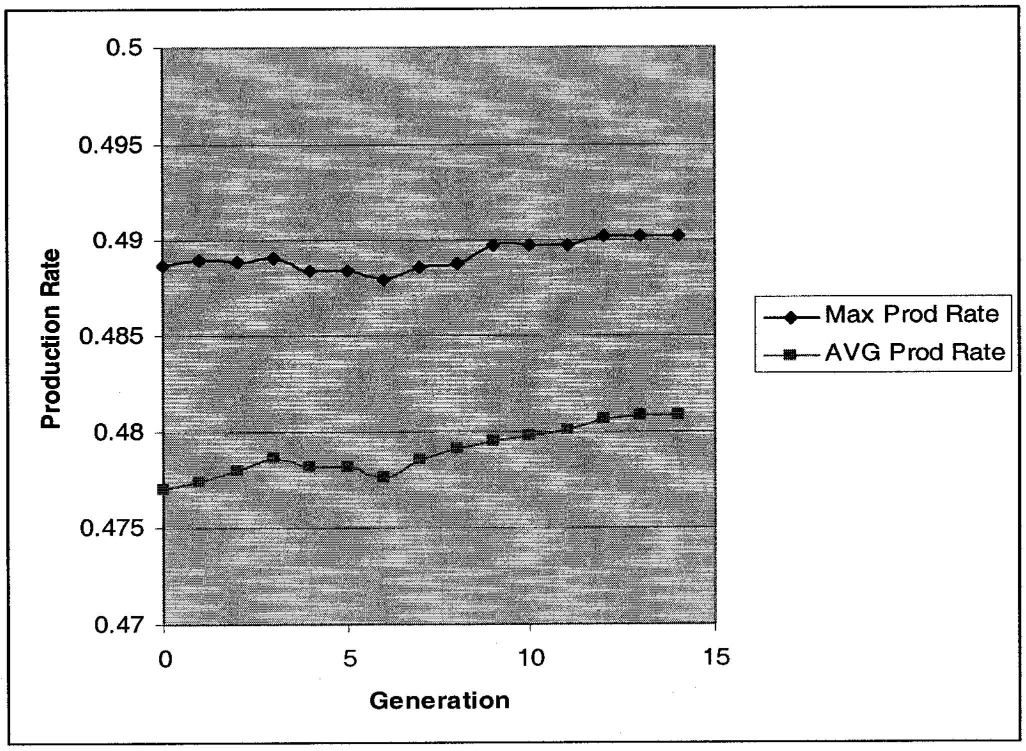


Figure 2A Convergence Behavior of Maximum and Average production rate (Table 2) of 7 stations Fork and Join Open Assembly System.

Table 3A: Iteration information for 11 stations Fork and Join Open assembly system with jam rates 0%, 3%, 0%, 2%, 0%, 0%, 3%,2%, 0% and 0% for stations 1 to 10 respectively.

Total Number of Pallets = 40

Generation	Buffers										MAX Prod Rate	AVG Prod Rate
	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10		
0	4	8	5	9	3	3	4	3	1	1	0.4319	0.4207
1	4	8	3	8	2	4	3	3	2	3	0.4321	0.421
2	4	8	4	7	2	4	3	4	2	2	0.4323	0.4216
3	3	8	4	6	1	5	4	4	2	3	0.4327	0.4218
4	3	8	4	6	1	5	4	4	2	3	0.4327	0.4219
5	3	9	3	5	2	4	5	4	3	2	0.4322	0.4226
6	3	8	3	5	2	5	4	5	3	2	0.4324	0.4218
7	3	8	5	6	1	4	5	5	1	2	0.4331	0.4223
8	3	7	5	5	2	5	4	7	1	2	0.4338	0.4227
9	3	7	5	5	1	5	4	7	2	2	0.4344	0.423
10	3	6	5	6	2	4	6	7	2	2	0.4348	0.4234
11	3	7	4	6	2	5	6	6	2	1	0.4351	0.4239
12	3	7	4	6	2	5	6	6	2	1	0.4351	0.424
13	3	7	4	6	2	6	5	6	2	1	0.4351	0.4244
14	3	7	4	6	2	5	5	4	3	2	0.4357	0.4248
15	3	7	4	6	2	5	5	4	3	2	0.4357	0.4248
16	3	7	4	6	2	5	5	4	3	2	0.4357	0.4248
OPT	**3**	**7**	**4**	**6**	**2**	**5**	**5**	**4**	**3**	**2**	**0.4357**	**0.4248**

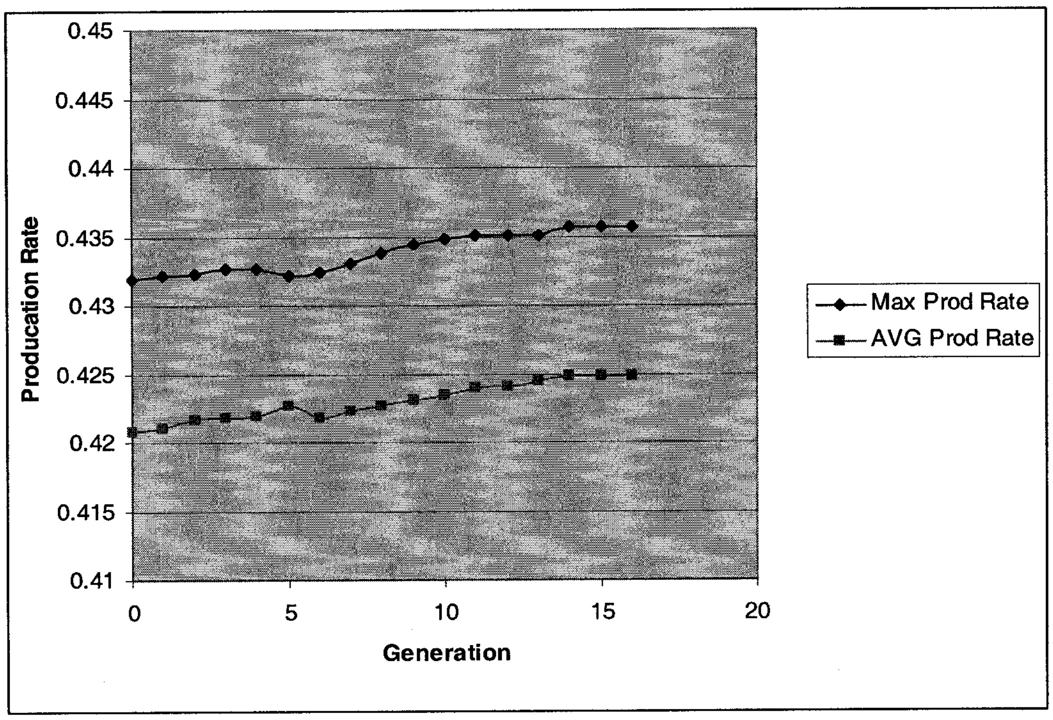


Figure 3A Convergence Behavior of Maximum and Average production rate (Table 3) of 11 stations Fork and Join Open Assembly System.

Table 4A: Iteration information for 11 stations Fork and Join Open assembly system with jam rates 0%, 2%, 0%, 2%, 0%, 2%, 2%,2%, 2% and 0% for stations 1 to 10 respectively.

Total Number of Pallets = 40

Generation	Buffers										MAX Prod Rate	AVG Prod Rate
	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10		
0	2	5	3	6	2	4	8	4	4	2	0.441	0.4286
1	2	4	2	5	1	4	7	6	7	2	0.4415	0.4289
2	2	4	2	5	1	4	7	6	7	2	0.4415	0.4294
3	2	4	1	5	2	6	7	5	7	2	0.4418	0.4298
4	2	4	1	4	2	6	8	6	5	2	0.4421	0.4301
5	1	6	2	4	2	5	7	6	6	1	0.4423	0.4301
6	1	6	2	4	2	5	7	6	6	1	0.4423	0.4304
7	1	3	2	6	2	7	6	5	6	1	0.4419	0.4305
8	2	3	2	5	2	6	7	5	6	1	0.4424	0.4307
9	1	4	3	4	3	5	6	5	7	1	0.4426	0.4307
10	1	4	3	4	3	5	6	5	7	1	0.4426	0.4311
11	2	4	3	3	4	6	5	4	6	3	0.4429	0.4315
12	2	4	2	4	5	5	6	4	6	1	0.4431	0.4319
13	2	4	2	3	4	6	7	4	5	2	0.4435	0.4317
14	2	4	2	3	4	6	7	4	5	2	0.4435	0.4321
15	2	4	2	3	4	5	7	4	7	2	0.4439	0.4324
16	2	4	2	3	4	5	7	4	7	2	0.4439	0.4327
17	2	4	2	3	4	5	7	4	7	2	0.4439	0.4328
18	2	4	1	3	3	6	7	4	6	3	0.4437	0.4329
OPT	**2**	**4**	**2**	**3**	**4**	**5**	**7**	**4**	**7**	**2**	**0.4439**	**0.4329**

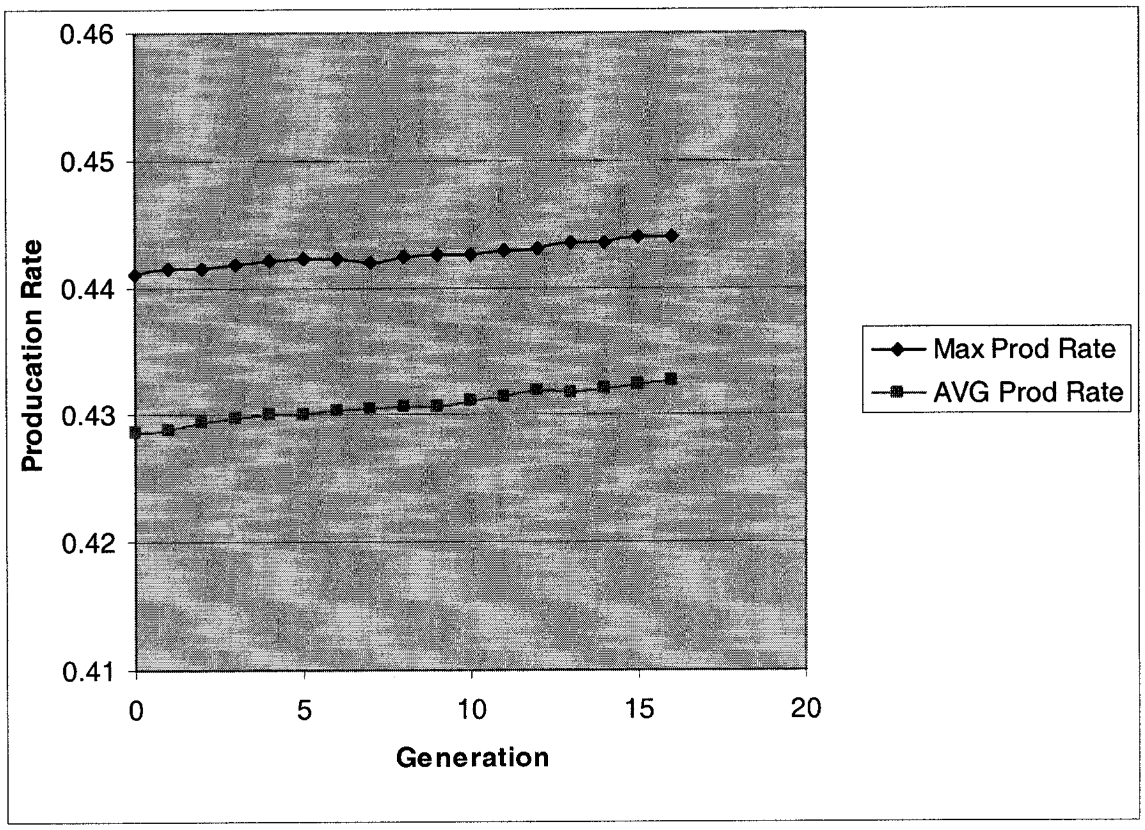


Figure 4A Convergence Behavior of Maximum and Average production rate (Table 4) of 11 stations Fork and Join Open Assembly System.

Appendix II: Convergence behavior of Max and Avg production rate for different configurations of Fork and Join Open assembly systems via Metamodeling-GA approach.

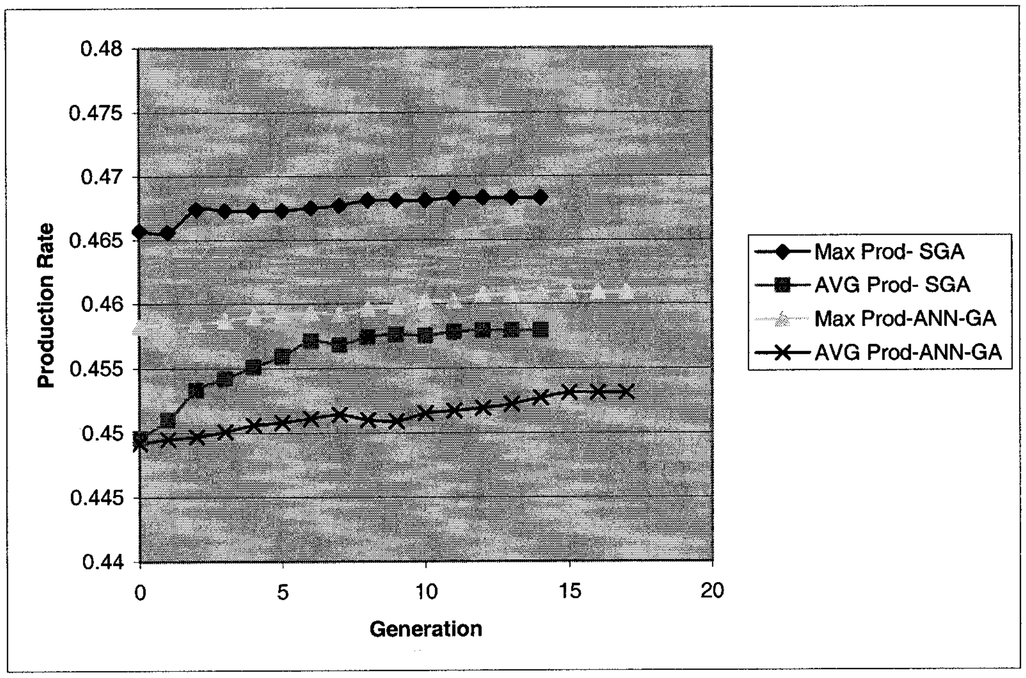


Figure 5A Comparison of ANN-GA and SGA Approaches for 7 stations Fork and Join Open assembly system with jam rates 0%, 3%, 2%, 0%, 2% and 0% for stations 1 to 6 respectively.

Total Number of Pallets = 30

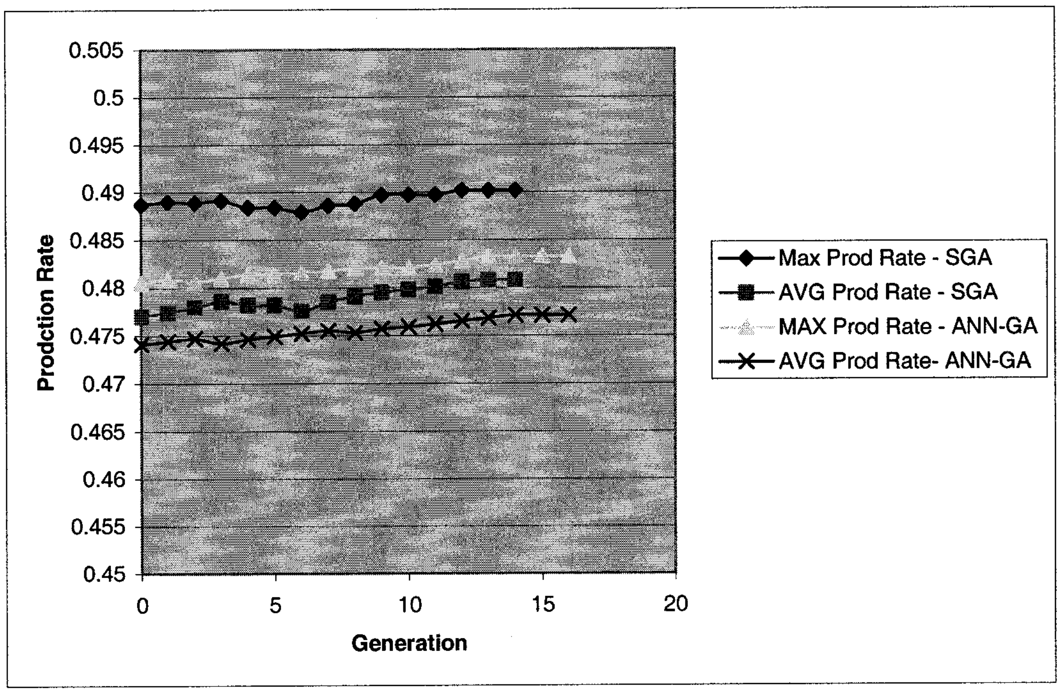


Figure 6A Comparison of ANN-GA and SGA Approaches for 7 stations Fork and Join Open assembly system with jam rates 0%, 2%, 3%, 0%, 3% and 3% for stations 1 to 6 respectively.

Total Number of Pallets = 40

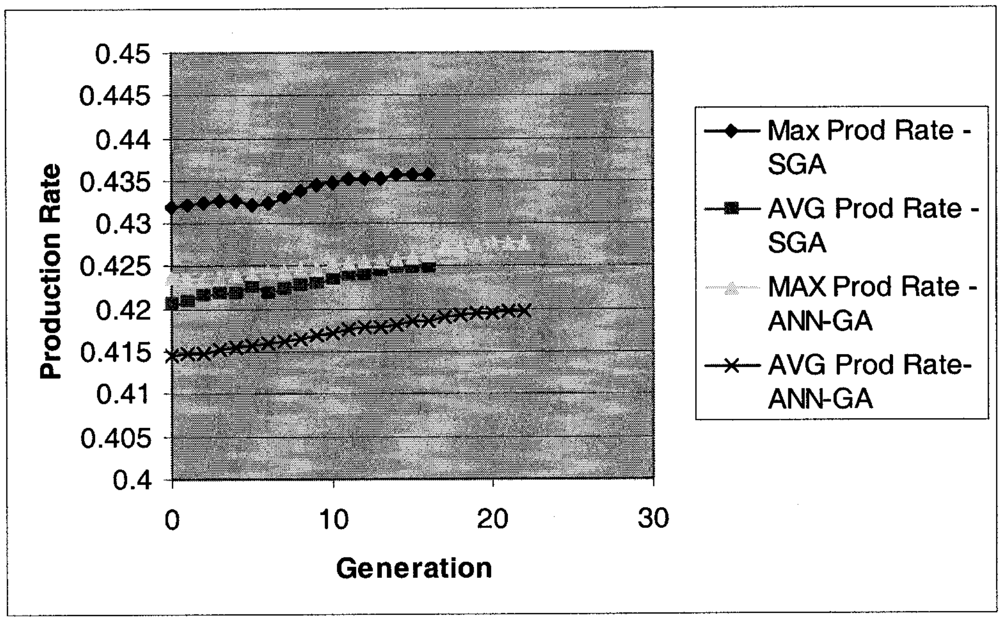


Figure 7A Comparison of ANN-GA and SGA Approaches for 11 stations Fork and Join Open assembly system with jam rates 0%, 3%, 0%, 2%, 0%, 0%, 3%,2%, 0% and 0% for stations 1 to 10 respectively.

Total Number of Pallets = 40

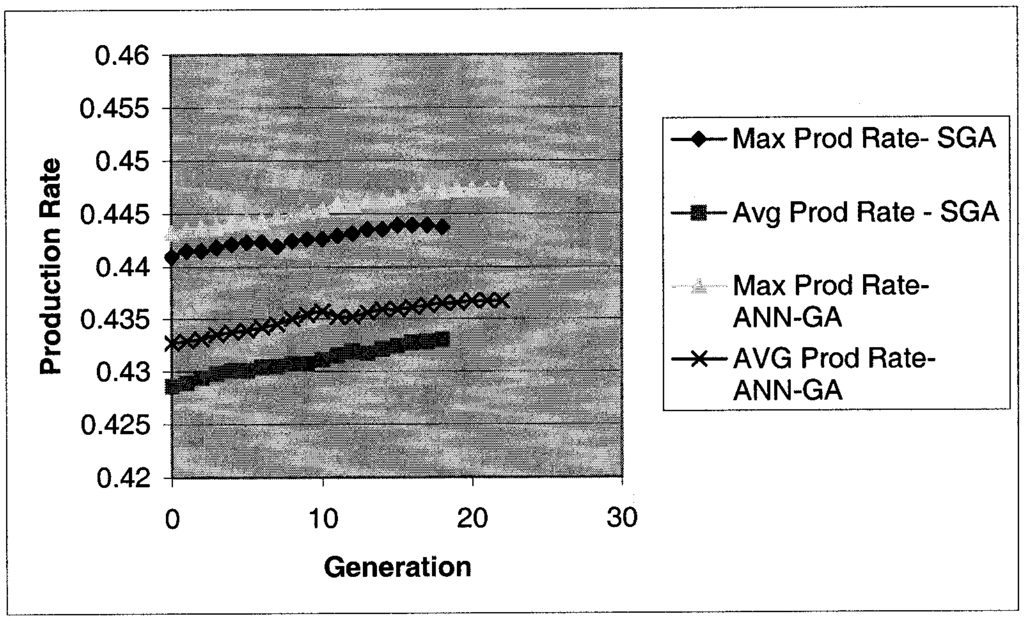


Figure 8A Comparison of ANN-GA and SGA Approaches for 11 stations Fork and Join Open assembly system with jam rates 0%, 2%, 0%, 2%, 0%, 2%, 2%,2%, 2% and 0% for stations 1 to 10 respectively.

Total Number of Pallets = 40

Table 5A Optimal Buffer Configuration by SGA and ANN-GA Approaches for 7 Stations Line

System No	Approach	b1	b2	b3	b4	b5	b6	Max Production Rate
System 1	SGA	3	4	7	4	8	4	0.4683
	ANN-GA	3	4	7	4	8	4	0.4609
System 2	SGA	4	10	6	3	9	8	0.4902
	ANN-GA	4	9	6	4	9	8	0.4831

Table 6A Optimal Buffer Configurations by SGA and ANN-GA Approaches for 11 Stations Line

System No	Approach	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	Max Production Rate
System 3	SGA	3	7	4	6	2	5	5	4	3	2	0.4357
	ANN-GA	3	6	4	6	3	5	5	4	3	2	0.4276
System 4	SGA	2	4	2	3	4	5	7	4	7	2	0.4439
	ANN-GA	2	4	2	3	4	6	7	4	6	3	0.4473