DEVELOPMENT OF DEMAND FORECAST AND INVENTORY MANAGEMENT DECISION SUPPORT SYSTEM USING AI TECHNIQUES.

Thesis

Submitted in partial fulfilment of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

by

PRASANNA KUMAR

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,

SURATHKAL, MANGALORE -575025

June, 2018

Dedicated to

My Dear Parents,

Beloved Wife

And

Loving Children

DECLARATION

By the Ph. D. Research Scholar

I hereby declare that the Research Thesis entitled ***"DEVELOPMENT OF DEMAND FORECAST AND INVENTORY MANAGEMENT DECISION SUPPORT SYSTEM USING AI TECHNIQUES"*** which is being submitted to **National Institute of Technology Karnataka, Surathkal** in partial fulfillment of the requirements of award of the degree **Doctor of Philosophy** in **Department of Mechanical Engineering** *is a bonafide report of the research work carried out by me.* The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

Register Number	**: 100469ME10P02**
Name of the Research Scholar	**: Prasanna Kumar**
Signature of the Research Scholar :	

Department of Mechanical Engineering

Place: NITK- Surathkal

Date:

Acknowledgement

I extend my warm and sincere gratitude to **Dr. Mervin A. Herbert,** Department of Mechanical Engineering, for all his valuable guidance, help and encouragement throughout this work. It has been indeed a great honor for me to work under his guidance and advice.With deep sense of gratitude and humility, I express my sincere thanks to him for his continuous support, motivation and inspiration, which made the research not merely educational but also enjoyable.

My sincere thanks to Mr. Gururaj, director of Valve chem Industries, Mumbai, who has given me an opportunity to study the Inventory Management System in his organization and provided me with the real time industrial data for my research work without which this doctoral work would not have been possible.

My genuine gratitude to Dr. Shrikanth Rao, who has been the driving force behind me and my thesis. My heart felt thanks to him for his initiative, continuous technical support.

I also take this opportunity to thank the Director, NITK Surathkal and Head of Mechanical Engineering Department, NITK Surathkal for allowing me to carry out my doctoral studies.

I sincerely thank the Research Progress Assessment Committee consisting of Dr. Vijaya Desai (Department of Mechanical Engineering), Dr. Shri Hari (Department of Civil Engineering), Dr. Mervin Herbert for their valuable comments and constructive criticism which have helped the enrichment of this doctoral work.

I am immensely indebted to the unending help and support I received from my co-research colleagues Mr. Subramanya, Mrs. Rashmi, Miss Charita, Mr. Karthik, Mr. Vignesh, Dr. Arun Kumar and Dr. Nagraj Shetty during the course of my research work.

I am indebted to my parents for inculcating in me the right values and virtues. I am extremely grateful to my beloved wife Jyothi and lovely children Vignesh and maneesha for enduring a lot of hardships caused on the domestic front due to my research priorities.

I wish to express my special thanks to all my family members and friends who were a constant source of motivation and encouragement during the entire course of my doctoral work.

PRASANNA KUMAR

Contents

List of Figures

List of Tables

Nomenclature

AI	Artificial Intelligent
ANN	Artificial neural networks
GA	Genetic algorithm
ACO	Ant colony optimisation
NP	Non deterministic Polynomial time
RBFNN	Radial basis function neural network
IM	Inventory management
OA	Orthogonal Array
MLP	Multi layer perceptron
MSE	Mean squared error
MAPE	Mean Absolute Percentage Error
DF	Degree of Freedom
Seq SS	Sequential Sum of Square
Adj SS	Adjusted Sum of square
F	Fisher’s ratio
P	Probability
Pred	Predicted
R Err	Relative Error

NOTATIONS USED

c	number of items
p	number of periods
$R_{u,v}$	demand rate of the item u at period v
G_u	ordering cost of the uthcomponent at the beginning of an interval
$O_{u,v}$	ordering quantity of uth component in interval - main decision variable
w	number of price discount breakpoint
$b_{u,w}$	wthdiscount breakpoint of uth component w=1,2,…….,w ($b_{u,1}$=0)
T	total storage space
T_u	required warehouse space per unit of the uthcomponent
$C_{u,w}$	purchasing cost of the uthcomponent at the breakpoint 'w'
C_u	purchasing cost of uthcomponent paid at the start of the interval
OC	total ordering cost
PC	total purchasing cost
HC	total holding cost
IC	total inventory cost
A	total budget
Z_1	upper band for O_{uv}
$I_{u,v}$	initial inventory of the component 'u' in interval 'v'
$B_{u,v,w}$	a binary decision variable; set equal one if component c is purchased at price breakpoint w in period v, and zero otherwise
$Q_{u,v}$	a binary decision variable; set equal one if a purchase of a component u is made in period v, and zero otherwise
M	total warehouse space
m_u	warehouse space for uth component

Chapter 1

INTRODUCTION

1.1 GENERAL BACKGROUND

Effective and efficient inventory management is vital for the organization to enhance functional efficiency across competitive business, to boost customer service and to improve the inventory cost effectiveness at different points of a supply chain network (Burgin et al. 1967). Critical decision which the companies are to implement on a regular basis is the procurement of products and raw materials. Errors in inventory decisions can lead to over stock that are expensive to maintain or understock which would lead to reduction in customer satisfaction level.

Inventory management decisions have to be arrived at under uncertain demand situations. They are also characterized by optimization of multiple objectives, most of the times, conflicting objectives like cost and service level (Brahimi et al. 2006).

The inventory control mechanism is the key issue in the field of industrial engineering and operational research (IE/OR) and still a green area in spite of several studies around this subject. As a crucial activity for any organisaton, inventory planning attempts to frame the decisions on procurement lot size and timing of stock replenishment. A common approach is a continuous-review (r,Q) reorder mechanism in which a procurement order of lot size Q is placed whenever the stock level gets reduced to the reorder point, r (Chopra et al. 2001). Lead time, historical fluctuation in lead time and the variation of demand govern the determination of (r, Q) so that inventory cost is minimized and customer service is maximized. Cost reduction and service level optimization are mutually incompatible goals which conflict with each other, as increasing the service level would require no stock out under any condition which can be achieved with only high inventory. Thus inventory decisions involve multi objective optimization strategies (Silver et al. 1990).

Also inventory management involves decision making under imprecise and uncertain demand, lead time and inventory cost. Fuzzy logics can be used to model these inherent uncertainties in the decision parameters so that decision environment comes as close as possible to real life situations (Du. T.C. et al. 1997).

Artificial intelligence techniques and tools such as neural networks, Genetic Algorithm, Fuzzy Logics, Ant Colony Optimisation provide a robust platform to deal with such uncertainty, imprecise details and multi objective optimization (Fei-Long Chen et al. 2010).

1.2. BACKGROUND OF DEMAND FORECAST

Demand forecast is an evaluation of anticipated future demand. A forecast can be estimated by mathematical means based on past data or it can be obtained by the subjective inferences of highly informed sources like subject matter experts. Sometimes the judicial combination of objective and subjective estimate makes the best forecast. A judicious hybridization of above procedures can also be adopted for demand forecast (Chandra et al. 2005).

Demand Forecast is essential for (Chen. F. et al. 2000):

- Future planning by reducing the effect of uncertainty.
- Anticipating in advance and effective change management.
- Improved communication and integration of planning teams.
- Balancing the demand fluctuation, capacity loading, inventory stock outs and delayed deliveries.
- Incorporating the operation cost variations into budgeting process.

- Enhancing the productivity and competitiveness by cost optimization, quality performance and higher level of customer satisfaction.

General methods of forecasting include qualitative techniques which are based on subjective judgment of experts in the field regarding future product demands and quantitative methods which are based on either time series or casual methods (Crum,C et al. 2003).

Recently there has been an increase in the research interests on the application of machine learning techniques such as neural networks for the demand forecast (Tugba Efendigil et al. 2009).

1.3 BACKGROUND OF INVENTORY MANAGEMENT

Inventory is the physical stock of goods kept for the purpose of future use. The term is generally used to indicate raw materials, work in process stock and finished goods, packaging, spares which are stored for meeting an anticipated demand in the future (Handfield et al. 2009).

Inventory management is the function of controlling the movement of goods through each and every component of supply chain from the procurement of the raw material to the inventory of finished goods in a systematic process to achieve the conflicting goals of maximum customer satisfaction with minimum cost and efficient operation (Javad Sadeghi et al. 2014).

The purposes of inventory are (Harris,F.W. 1990):

- To handle fluctuation in product demand without delayed deliveries or lost sales.
- To have increased flexibility in production scheduling to take advantage of better capacity loading and reduced machine loading time.
- To ensure smooth production even in case of delayed delivery from vendor side.

- To capitalize on optimum purchase order quantity.
- To maintain buffer in the production line through in process inventory to avoid independence of operations.

Many inventory control models have been proposed to effectively manage inventory minimizing the total cost and maximizing the service level to customers. Several AI techniques have been used to model the uncertainties involved in inventory management like demand, lead time, inventory cost etc.

Optimisaton of conflicting objectives in inventory management using the different AI approaches is a field of study which has attracted lot of research interests [Jui-Tsung Wonga et al.2011, Hui-Ming et al. 2009, Kazemia et al. 2010].

1.3.1 EOQ model

It is one of the oldest, simple inventory control models. Ford W. Harris in 1913 has suggested this model for the first time (Harris, F.W., 1990). It is an important deterministic continuous review model. As the ordering quantity increases, inventory holding cost also increases but ordering cost comes down due to lesser number of orders. Economic order quantity is that optimum order size which minimizes the total inventory cost. The total inventory cost generally consists of two main components, carrying cost and ordering cost.

The EOQ model works under the assumption that the demand for the product is constant over the year and that each order is supplied in entirety when the stock level falls to zero. Each order will incur a fixed cost irrespective of number of units ordered (Krone, L., 1964).

The optimum number of units in each order has to be determined so that total cost related to purchase, delivery and storage of material is minimized.

The data required for the solution of the EOQ model are (Leender et al. 1985):

- Annual demand for the product.
- Fixed cost incurred for placing each order.
- Purchase cost of item.
- Storage or carrying cost for each item.

Following basic assumptions are the basis for computing EOQ (Phillips, D. et al. 2006).

- The demand rate is deterministic, and uniform during all the periods of the year.
- The ordering cost is fixed.
- The lead time is constant.
- No discounts in purchase price are considered.
- The stock renewal is made instantly and full batch of the items is delivered immediately.
- No multiple products are involved.

EOQ is the order quantity for minimizing (ordering cost + carrying cost).

- P = Procurement Price.
- Q = quantity ordered.
- Q^* = Economic order quantity.
- D =yearly demand.
- S = Ordering cost.
- H = yearly carrying cost per unit or holding cost (Cost incurred for warehousing, cold storage, insurance etc. usually are not a part of the unit cost).

The following cost function is minimised for obtaining the Economic order Quantity:

Total Cost = Purchase cost + Ordering cost + Holding cost   (Silver et al. 1990).

Purchase cost: This is the cost to purchase the item. It is variable and equal to the product of unit price and annual demand quantity.

Ordering cost: This is the cost incurred for placing orders. Each order incurs a fixed cost S. Total ordering cost is the product of S and the number of orders. The number of orders/ year = D/Q. Total ordering cost is S × D/Q.

Holding cost: Total holding cost is the product of per unit holding cost and the average stock during specified horizon. In this case, average stock is

(initial inventory + final inventory)/2

$$\mathrm{TC} = PD + \frac{DS}{Q} + \frac{HQ}{2} \qquad (\text{Harris, F.W., } 1990).$$

Minimum value of the function is obtained by equating the partial differentiation with respect to Q to zero. (based on assumption that all other variables are constant)

$$-\frac{DS}{Q^2} + \frac{H}{2} = 0$$

Solving for Q gives Q* (the optimal order quantity):

$$Q^2 = -\frac{2DS}{H}$$

$$Q^* = \sqrt{\frac{2DS}{H}}$$

Therefore:

Q* is independent of P; it is a function of only S, D, H.

Fig 1.1 graphically represents the relationship between the different cost components, and relationship between cost and order quantity size.

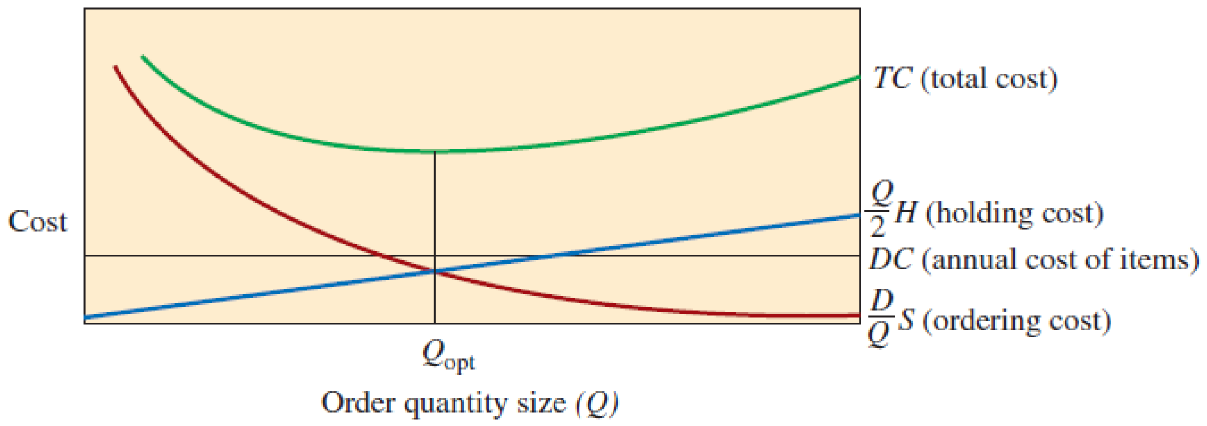


Fig 1.1 Annual cost based on size of order (Courtesy: Silver et al. 1990)

The simple EOQ model can be made to represent more practical situations with the addition of several constraints like back ordering costs, Quantity discounts, price breaks and multiple items.

1.3.2 SAFETY STOCK

Safety stock (also called buffer stock) denotes a level of additional stock that is held to alleviate risk of stock out due to uncertainties in demand level and delay in delivery. Optimum safety stock levels enable business continuity without interruptions due to stock out. Safety stock insures the business operations against stock outs. Average historical variations in demand level and lead time would provide the required data to fix the safety stock for each item (Tan et al. 2000).

When the company introduces a new product, safety stock can be used as a tactical tool until a few months or weeks of operation, when the marketing department can come up with more accurate demand forecast. When demand cannot be forecast with precision, higher safety stock is required to maintain a higher service level to customers. However, appropriate business approach is to minimize the safety stock level to reduce the locked up capital when the product demand prediction gets better. For companies which adopt lean manufacturing and for those who have very limited financial cushion this becomes a crucial business strategy to reduce the safety stock to minimum (Chang et al. 2001).

The amount of safety stock which a business entity opts to hold, can have a large scale impact on their bottom line. Excess safety stock causes increase in inventory holding costs. In addition, when the items are in the inventory for a long time, there is a chance of product quality getting deteriorated, validity getting expired, or getting out dated which will increase, considerably, the inventory cost. Less than optimum safety stock may make it difficult to service the customers specially when there is high demand and when there is heavy supplier lead time fluctuation. Dissatisfied customers are heavy on balance sheet of organization. As a result, finding the precise trade off premise between excess and insufficient safety stock is very crucial for the organization (Chopra et al. 2001).

Organizations adopting ‘make to stock’ business strategy find the concept of safety stock very useful and justified to support their business operations. When the lead time from supplier side is too long, this strategy is used by the organizations so that they can deliver to their customers from the stock without the impact of delayed deliveries of the raw materials from their suppliers.

The other foremost objective of safety stocks is to absorb the variation of the product demand level. The predicted demand is the basis for the production planning. But actual demand scenario may be far different based on the prediction accuracy or changing situations. Safety stock would bridge the gap and avoid lost sales keeping high customer service level. It will also guard against the effects of any unforeseen interruptions like machinery failure or delayed supplies. Fig 1.2 depicts the importance of safety stock in the inventory model.

•

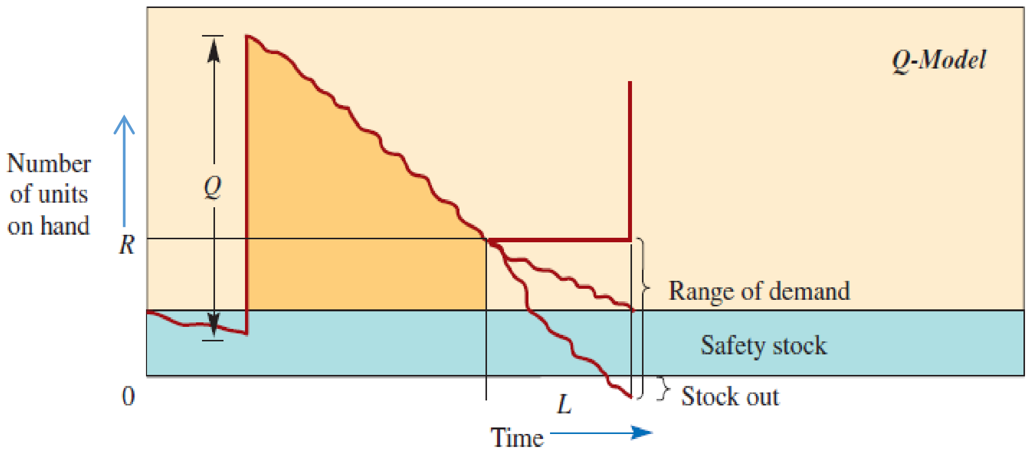


Fig. 1.2 Fixed order quantity model with safety stock (Courtesy: Lee et al. 1997)

1.3.3 Reducing safety stock

Safety stock is utilized as a safeguard to defend organizations from ill effects of zero inventory level caused by imprecise planning or delayed deliveries by suppliers. Generally cost of inventory is perceived as a burden on financial bottom line of the organisation. In addition, perishable goods like food and drink, could get spoilt and cannot be stored beyond their shelf life. Organizations can make use of several approaches to minimize the safety stock. Achieving accurate prediction of demand is one of them. Increased collaboration with suppliers to minimize lead time fluctuation and guarantee on time delivery is another approach. Company which adopts lean supply strategy, tries to reduce the lead time. Lead time reduction would help to minimize safety stock levels (Christopher, 1992).

To have a tradeoff between service level and safety stock, the organizations practice safety stock calculation based on their opted service level (Cox, A. et al. 2001). Just as an illustration, an organization can specify that its safety stock requirement is for a service level of 95%, which means that 95 out of 100 times, the company would be able to serve the customers with the help of its safety stock, and this is satisfactory for the company. Higher than this level, would not be cost effective for the company. The lower the service level company settles for, lower will be the safety stock requirement.

An efficient Enterprise Resource Planning system (ERP system) can effectively assist an organization in framing their inventory management policy so as to minimize its buffer of safety stock. Most ERP systems are equipped with a well organised Production Planning module. This module enables an efficient demand prediction system so that company can work with a highly accurate and dynamic sales forecasts. Precise and dynamic forecast would reduce the probability of stock out of raw material or inadequate inventory of finished products. This would drastically reduce the amount of safety stock that the company should maintain for the same service level. In addition, ERP systems recommend proven formulas to help compute the required levels of safety stock based on their experience in similar industries and similar business conditions. While an ERP system helps an organization in assessing a reasonable amount of safety stock, the ERP module must be designed and implemented to plan requirements effectively and efficiently (Gilbert, K. et al. 2005).

1.4 BACK GROUND OF ARTIFICIAL INTELLIGENCE

Artificial Intelligence is a comprehensive concept covering a variety of disciplines and applications like machine learning, natural language processing, pattern-matching, and expert systems (Rolston, D.W. 1988). In the year of 1956, John McCarthy coined the word and idea of Artificial Intelligent system. Artificial intelligence is the computational model of human behavior.

AI techniques, also known as soft computing is a collection of distinctive approaches, comprising mainly of Neural Networks (NN), Expert System (ES), Fuzzy Logic (FL), and Evolutionary Algorithms (EA), which offer flexible information handling and processing capabilities and human thinking approach to solve real-life problems. They are intelligent agents and solve real life problems which are hard to be simulated as mathematical model (Mark Ko et al. 2010).

1.4.1 Fuzzy logic

Fuzzy set theory was initially proposed by Lotfi Zadeh (Zadeh Lotfi, 1965). It provides a basis for framing mathematical model to deal with and characterize uncertainty, ambiguity vagueness, imprecision, fractional truth, and deficiency of information (A. Tettamanzi et al. 2001). As the fundamental concept of artificial intelligence, fuzzy logic provides computational capability for the simulation of the thought and perception processes. Fuzzy systems are beneficial in conditions involving highly complex systems and in scenario where exact solution is too costly to achieve and that is why approximate solution at lesser cost is justified (Ross, T.J.2004). To represent and treat qualitative, approximate, indeterminate and complex processes, the fuzzy logic system can be well implemented since it displays a human like thinking process. In contrast to on-off logic, fuzzy logic utilizes multi valued logic concept to simulate approximate reasoning (Du. T.C et al. 1997).

1.4.2 Artificial Neural Network

Neural network is a collection of nonlinear processing units called neurons which are capable of parallel processing. They have distributed information handling and processing structure. Neural network have the capability to accept the input and to function as mathematical processor executing the specified operations to produce the output (P. Musilek et al. 2000). They can also identify the pattern from the given data and then complete the next data range by simulating the human brain process of reasoning and pattern matching, They can be trained in eliminating noisy data and in retrieving correct information by duplicating human brain process.

In terms of modeling, significant research work has been carried out in the recent past to develop the capability of artificial neural networks (ANN). In artificial neural networks, neurons are powerfully interlinked. By themselves, they exhibit simple behavior, but when connected they are powerful enough to solve complex problems (Kartalopoulos, 1996).

1.4.3 Evolutionary algorithms

Evolutionary algorithms (EA) were conceived to simulate some of the processes which are evident in natural evolution. In the natural evolution process, chromosomes are the organic systems which are responsible for encoding the structure of living beings. Failed structure chromosomes do not reproduce more, but chromosomes which represent successful structures reproduce more frequently. This is ensured by the natural evolution process. This process of survival of the fittest is utilized to solve many complex real life situations by using simple encoding and reproduction mechanisms (Davis, 1991).

One of the distinctive subclass of broader set of EA technique is Genetic Algorithm. GA was first introduced by John Holland (John Holland, 1975). Since then lot of research has been done on the application of GA for multi objective optimization (Tettamanzi, 1991). Wherever search and optimization problems are involved, GA has been used as robust solutions methodology with the advantage of adaptiveness and flexibility. In complex multi objective optimization problems, where the number of variables is very high and where traditional search and optimisations techniques have failed to yield a satisfactory result, GA has been gainfully adopted to obtain near optimal solutions using the simulation of some of the features of biological evolution. That is why, GA has attracted a lot of attention among research fraternity (Goldberg, 1989).

1.4.4 Ant colony Optimisation

Ant colony optimization (ACO) is a meta heuristic algorithm which is used for near exact optimization solution. It draws its working principle from swarm intelligence and was first introduced by Dorigo in the 1990s (Dorigo M. et al. 1994). It is one of the innovative approaches for the Multi objective optimization which is motivated by the real ants who search an approximately optimal path between their Nest and the food source, as shown in Fig. 1.3 [Ali Roozbeh Nia, et al. 2014, Colorni et al.1992]

During their food pursuing journey, ants accumulate chemical substances called pheromones on their return trip back to their nest. Next group of ants while searching for the food, are guided by the smell of pheromones and are attracted to the marked paths.

The higher the density of the pheromone that is deposited on a path, the more number of ants would follow that path. The pheromone vapors are evaporated over time. Evaporation reduces the pheromone concentration on extended and less interesting paths .Shorter paths are visited again and again, revitalized more rapidly, therefore having the chance of being repeatedly explored. Obviously, ants will connect themselves to the most efficient trail because it acquires the heavier density of pheromone (Ali Roozbeh Nia, et al. 2014).

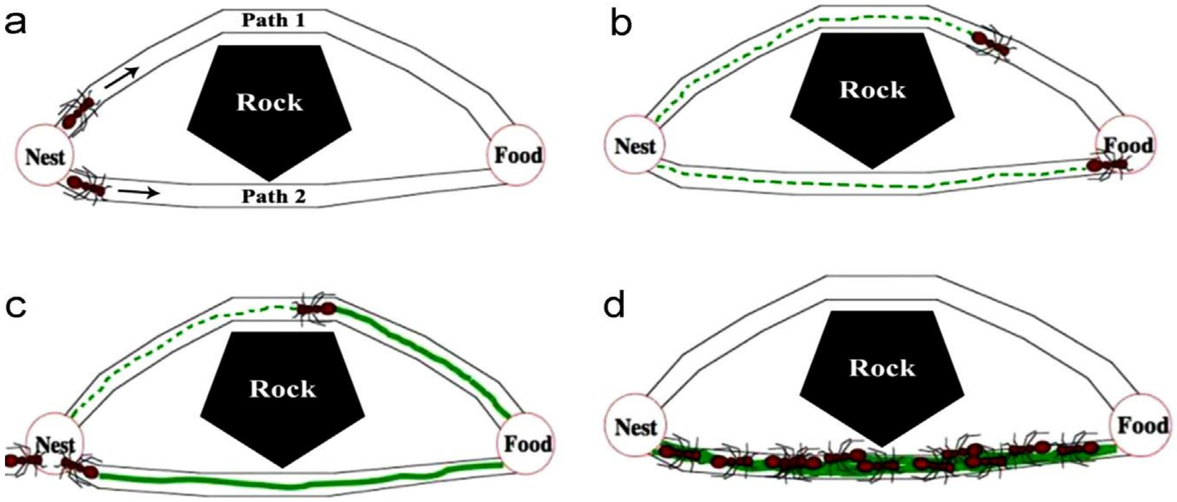
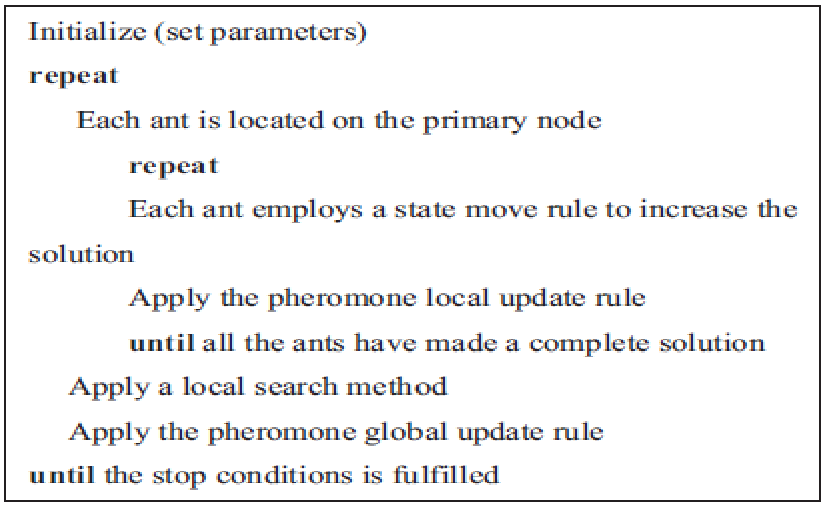


Fig 1.3 Basic Behavior of Ant Colony Optimization at different time periods. Concentration of pheromones on each path is represented by the green line. (Courtesy: Ali Roozbeh Nia, et al. 2014)

ACO algorithm simulates the above performance by generating pheromone concentration maximization. This is effected by two important operations.

- The quantity moderation of pheromone which determines the pheromone accumulation and drying up rate.
- The state transitional rule that is probabilistic in nature and picks up an end point depending on pheromone concentration. (Colorni et al. 1992, 1994).

Fig.1.4 shows the procedural steps of standard ACO algorithm.



```
Initialize (set parameters)
repeat
    Each ant is located on the primary node
        repeat
        Each ant employs a state move rule to increase the
solution
        Apply the pheromone local update rule
        until all the ants have made a complete solution
    Apply a local search method
    Apply the pheromone global update rule
until the stop conditions is fulfilled
```

Fig.1.4 The procedure involved in the ACO algorithm. (Courtesy: Ali Roozbeh Nia, et al. 2014)

1.5 PROPOSED WORK SUMMARY

Accurate demand forecast, efficient and effective inventory management are the crucial factors for an organization to compete in the marketplace. Traditional demand forecast methods and inventory management models suffer from severe limitations due to inability to process non linear data and meet the complexities of modeling and simulating real life situations. Over the past two decades, AI technology has emerged as an important development in the field of information science. So, it is proposed to study in detail the current practice of demand forecast and inventory management in the case of a valve manufacturing company which represents a semi make to order manufacturing industry. AI technique of neural network will be applied for demand forecast. Different architectures of neural network will be explored for achieving higher demand forecasting accuracy. The output of neural network demand forecast will be input for a novel GA inventory model for optimum lot sizing. Developed GA model will be compared and validated with the conventional inventory control model.

1.6 ORGANIZATION OF THE THESIS

In the present investigation, a systematic study is carried out on the two important aspect of supply chain management: demand forecast and inventory management. Data collection carried out from the researched valve manufacturing company which represent semi make to order industry. An integrated application of AI methodologies is carried out to improve demand forecast accuracy and optimize the lot size for a periodic review multi item, multi period inventory model. The various stages of the study and investigation are divided into six chapters. The summary of discussions carried out chapter wise is detailed below.

Chapter 1 elucidates on the historical background as well as the encounters tackled and inspiration to take up the present work. An overview of the proposed work with specific objectives is also articulated in this chapter.

Chapter 2 deals with a step by step detailed and critical review of literature in the area of demand forecast, inventory management. The importance of demand forecast and inventory management as important supply chain management functions have been reviewed. Traditional demand forecast techniques have been analysed in terms of advantages and limitations. Neural network application for the demand forecast has been discussed. Brief explanations of different inventory management models are discussed. The past and current research work related to application of different AI techniques is reviewed with more focus on multi item multi period periodic review lot sizing optimization. The mathematical model created for analysis, optimization and prediction in the field of demand forecast and inventory management is highlighted. Application of GA technique for the field of multi objective optimization is also emphasized. Gaps in the knowledge which has inspired the present work is listed. Clear objective for the present study is set and the elaborate scope is defined.

Chapter 3 deals with the research methodology. Block diagram representing the various steps of the planned research work has been furnished. Data collection methods that are adopted have been defined and explained. Sample data for demand forecast is shown. Detailed description regarding the application of AI tools-neural networks, Genetic Algorithm, Ant colony Optimisation to the present work has been elucidated. Taguchi Design of experiments to arrive at the optimum values of parameters for GA program has been explained.

Next 3 chapters are dedicated for the results and discussion in 3 parts. Chapter 4 elaborates on results of the present research work on ANN modeling of demand forecast. The demand predicted based on different ANN models have been compared with actual sales data and best architecture of ANN model has been identified. Further, the model has been validated based on the comparison of future predicted demand and actual sales data for that particular time period.

Chapter 5 and 6 discuss the ACO (Ant Colony Optimisation) modeling and GA modeling of inventory management respectively. Mathematical formulation of the Multi objective optimization GA and ACO model for multi period, multi item periodic review lot sizing has been explained in these chapters followed by the application the of the tool for the data sets obtained from real world industrial scenario of inventory control. The ACO and GA model have been validated and their performance compared based on the important parameters of objective function and CPU time of execution.

In Chapter 7, the overall conclusion derived from the present research work is elaborated and further direction of research work is presented.

Chapter2

LITERATURE

INTRODUCTION

Demand forecasting is one of the most important functions of organization. Accurate demand forecast keeps demand and supply in equilibrium. It has a great role in decreasing surplus and scarcity of inventories and enhancing the profitability (C.Crum et al.2003).

In a continuously shifting and highly competitive business environment, arriving at the correct decisions in right time based on demand forecast becomes mandatory for the organization (Tugba Effendigil et al. 2009). Accurate demand forecast is an important constituent of supply chain management and an important prerequisite for inventory management. An increase in forecasting accuracy will result in cost effectiveness because of optimized inventory (R. Carbonneau et al. 2008). It will improve the key performance indicator of customer satisfaction and on time deliveries.

Forecast is an evaluation of future values of specific quantified pointers relating to a business decision environment by analytically combining and extrapolating the data about the past. It is a numerical indication of the future trend. Upward misjudging of the demand forecast will prove very expensive for the company due to disproportionate inventory costs, forced price slash and squeeze in margins due to market glut, unwarranted increase in production and storage capacity and missed opportunity for the sale of higher margin products. On the other hand, downward bias for demand forecast would lead to inability to serve the market requirements and exploit the market opportunities, increased customer dissatisfaction, losing to competitors in market place (Ravi Mahendra Gor et al. 2010). Marketing and production managers need to comprehend well the significance of accurate demand forecast if they have to steer the company to high growth trajectory.

2.1 METHODS OF DEMAND FORECAST

Generally, forecasting methods can be classified into three classes. Different approaches are used under each class [Aburto et al. 2007, Weihui Deng et al. 2016, Chirag Deba et al. 2017, Maria Rosien kiewicz et al. 2017]. The main classification is as follows:

Qualitative approaches: Used mainly for planning of long term objective and goals and also for planning the major investment facilities decision.

Quantitative approaches: Used mainly in short term tactical decision making like production forecast, inventory control. Many analytical and mathematical tools like time series are used in this method.

Casual Technique approach: These methods are used in intermediate-term aggregate planning.

Fig 2.1 shows the various Forecasting techniques that can be used for demand prediction in an organization.

2.1.1 Qualitative techniques in Forecasting

Qualitative methods are characterized predominantly by subjective decision making. In this approach, more stress and reliance is on human judgment and ability to make an accurate opinion and extrapolation of business indicators. Tugba Effendil et al. (2009), Chandra C. et al. (2005), Dejonckheere et al. (2003), Cox A. et al. (2001) worked extensively on different qualitative technique approaches.

Chandra, C. et al. (2005) examined the qualitative demand forecasting method where the estimation is built based on the decisions and deliberations of the person who are at the grass roots or who are in close proximity to the end user. He established that the fundamental basis for this method is the fact that person close to customer or end user would be able to predict the future demand very accurately. Even though, this method cannot always be true, its basis is on valid assumption which makes it many times highly reliable

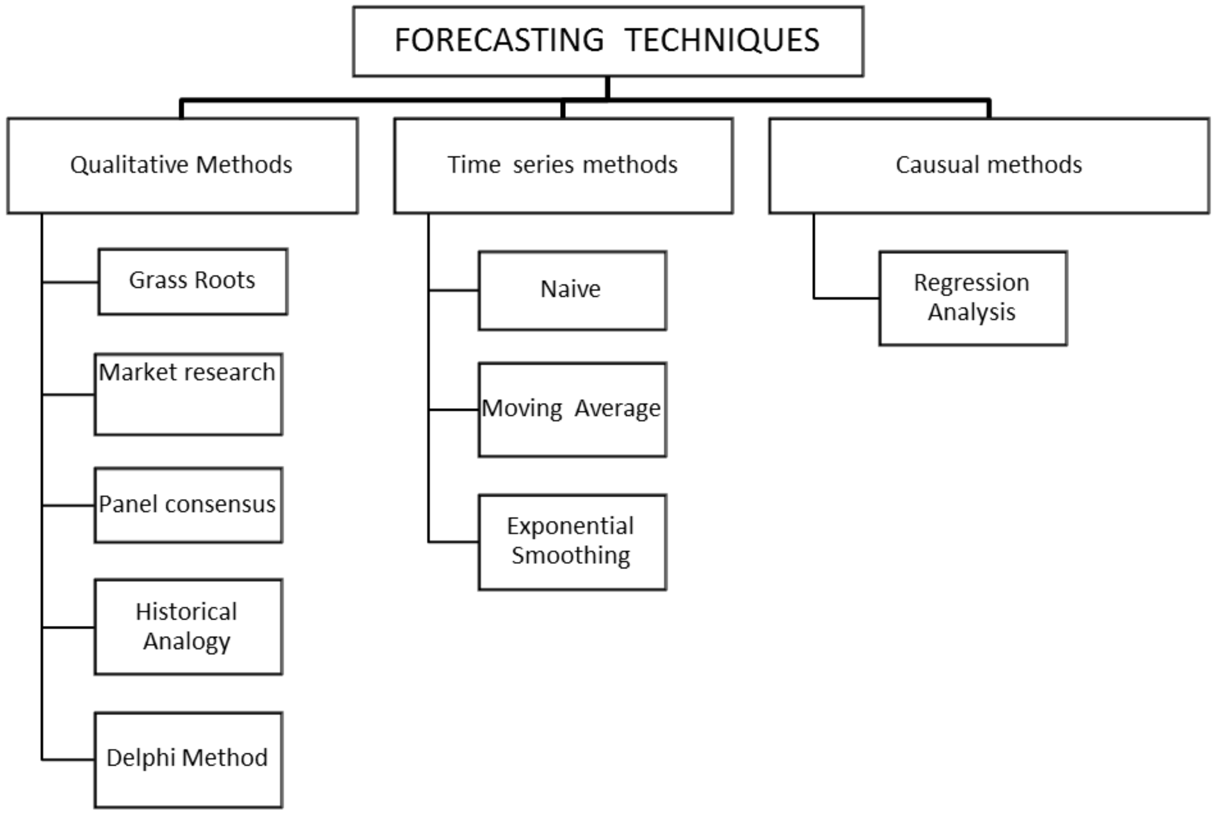


Fig 2.1 Forecasting techniques (Aburto et al. 2007).

Cox et al. (2001) refined the qualitative approach for demand forecast and used the panel consensus method. The basis of panel consensus method is the concept that two heads always think better than one. A panel of people from different positions and departments are consulted and a forecast is developed which is definitely more accurate than developed by a smaller group. It was established that unrestricted flow of information and ideas in open meetings is essential for the panel consensus to succeed.

Improvement on this method was suggested by the work of Dejonckheere J. et al. (2003), where the demand forecast was obtained by research made by specialists in this field, through different data collection methods like market survey and interviews. This type of survey is conducted mainly for development and testing of new product concepts, taste and distaste of current products and customer preference of particular product class. Tan, K.C. (2001) used Delphi method which is based on the response of a group of experts to a survey form set by a moderator which reflects their opinion about the demand. He compared it with other qualitative methods and arrived at the conclusion that Delphi method can yield highly reliable result within a reasonable time provided that optimum number of experts are contacted.

2.1.2 Causal Methods

Causal methods are based on the presumption that demand forecast is substantially linked with specific aspects in the external domain e.g., the financial position of the country, general business conditions, rate of interest. Lee, H. L. et al. (1997), Chopra, S. et al. (2001) and Chandra, C. et al. (2005) reported numerous research work carried out on this approach of demand forecast.

In his research studies, Lee, H. L. et al. (1997), primarily identified the relationship between dependent and independent variables and modeled the forecast based on the measured association. He used linear regression analysis which is one of the main casual methods, and determined the correlation between a dependent variable and one or more independent variables through mathematical formulation.

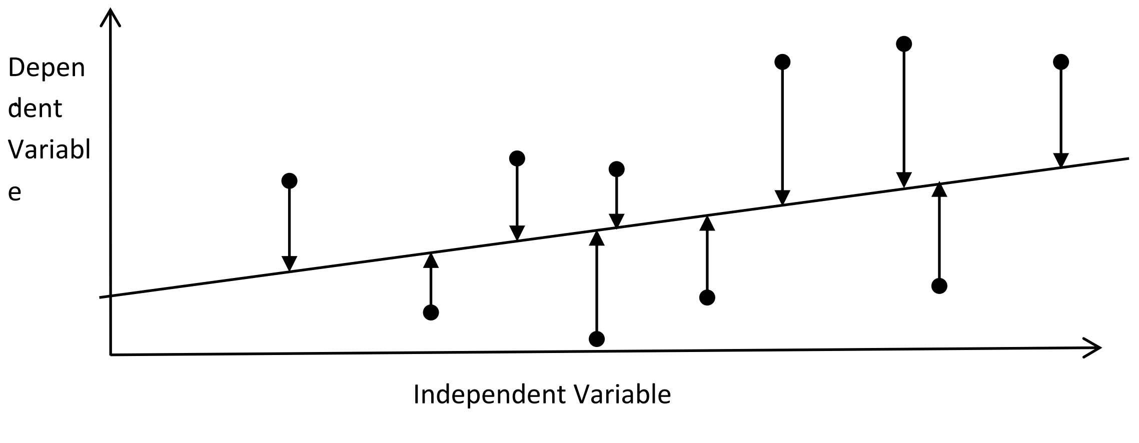


Fig 2.2 Linear Regression as a Casual Forecasting model (Chandra, C. et al. 2005)

Chopra, S. et al. (2001) further refined the process of establishing the relationship from data collected. Linear regression represents the distinct division of regression where the variables are linearly related. Fig 2.3 shows the application of linear regression as a Casual Forecasting model.

Chandra, C. et al. (2005) compared the technique of linear regression as applied for forecasting of time series and for casual relationship forecasting. In time series analysis, the dependent variable which is plotted on Y-axis is analysed with respect to time on X-axis. In the casual relationship forecasting, regression analysis develops the mathematical equation to describe the relationship between dependent and independent variable.

2.2 EVALUATING THE FORECAST ACCURACY

There are many approaches to evaluate prediction accuracy. Mean Absolute Error (MAE), the mean absolute percentage error (MAPE) and the mean square error (MSE) are the important indicators used to judge the forecast accuracy (Makridakis, S. et al. 1998).

Error = Actual Observed value – Predicted value.

Absolute Percentage Error = (Actual value - forecast value / Actual Value) × 100.

MAPE = the average of the Absolute Percentage Errors.

MSE = the average of the squared errors.

MSE and MAPE are defined as in following Equations [Tugba Efendigil et al. 2009, Lewis C.D.,1982].

$$\text{MSE} = 1/n \sum_{t=1}^{n} (O_t - F_t)^2 \qquad \text{Eq 2.1}$$

$$\text{MAPE} = 1/n \sum_{t=1}^{n} \left(\frac{O_t - F_t}{O_t}\right) * 100 \qquad \text{Eq. 2.2}$$

where O_t denotes the actual observed value for time period t and F_t represents the predicted value for the same time period, n is the number of periods.

2.3. TIME SERIES METHODS OF DEMAND FORECAST

Time-series techniques employ past data to build a forecast [Tugba Effendil et al. 2009, Aries et al. 2016, John E. Boylan et al. 2012, Van Wingerden et al. 2014]. Some of the examples of time series data are past periodic sales records of products, quantity requirement data records of services like telephone, power or transportation. Numerous standard approaches are available which are widely recognized as the time series methods. These models analyse past statistical data to workout forecasts for the future. The essential hypothesis here is that past correlations will continue to retain in the future (Boylan, J.E. et al. 2007). The various procedures differ largely in the way in which the past values are correlated to the predicted ones.

A time series corresponds to the historical recorded values of the variables under study. The values for the variable under study are collected for different time intervals as per the problem under study and depending on the analysis required. The periods may be very short like seconds or minutes or longer like days, weeks or months (Makridakis, S. et al. 1998). Following are different approaches to analyze the time series for demand forecast.

2.3.1 Naive methods

As the name suggests, naive forecasting method is simple and does not involve much mathematical calculation. Makridakis, S. et al. (1998), Gurani, H. et al.(1999), and R. Carbonneau et al. (2008) carried out detailed research into various aspects of Naive methods. Makridakis, S. et al. 1998 investigated the obvious and simplest form of naive forecast which assigns the most recently noticed value as the predicted value for the next periodic interval. Essentially, this approach to naive forecast depends only on the just previous observation and prior values are ignored. In another approach of naive forecast called free hand projection method, a free hand curve is fitted to represent the time series and forecast is done by extrapolation (Gurani, H. et al.1999). The free hand curve is extended to develop the forecast values of the time series. R. Carbonneau et al. (2008) compared naive method with other time series methods to highlight its limitations with respect to the achieved prediction accuracy.

2.3.2 Simple Moving Average Method

A simple moving average is calculated based on the formula:

$$V_t = \frac{o_{t-1} + o_{t-2} + o_{t-3} + \cdots + o_{t-n}}{n} \qquad Eq.\,2.3\ (\text{E.Bradley.}\,2003).$$

where, V_t = Forecast value for the next period, n = Periodic intervals under study, o_{t-1}, o_{t-2}, o_{t-3} and so on are the actual observed demand in the previous period, two periods before, three periods before and so on, respectively (E.Bradley. 2003). Investigative research reports on moving average methods have been presented by various analysts like E. Bradley (2003), G.E.P. Box et al. (2004), Hill T. et al. (2006).

E. Bradley (2003) studied the conflicting requirements of larger period lengths on prediction accuracy. For higher forecasting accuracy, it is essential that the best period of moving period is chosen. Longer moving average period will help in smoothening out or averaging out the random elements. Main limitation of moving average method is the undesirable characteristic of trailing the trend. Shorter time span has the disadvantage of larger fluctuation, but at the same time, has the advantage of closely following the trend. Hill T. et al. (2006) investigated the effect of extensive randomness with underlying trend in the data pattern which would necessitate greater number of periods of moving average. He concluded that higher number of intervals in the moving average will increase the smoothing effect and regulate the fluctuations. Box et al. (2004) through his research work suggested that giving more weightage to recent data would definitely increase the forecasting accuracy as it would more closely reflect the recent conditions. He also quantified the effects of not considering all the past data for the moving average calculation by comparing the moving average of different periods.

2.3.3 Weighted Moving Average

The limitation of equal weightage for all the data in the moving average calculation is overcome by differential weightage assigned to the different data in weighted moving average method. Any fractional weights can be placed on the individual data depending on its importance in prediction performance, subject to condition that sum of all the weights is unity (Kuo C. et al.1995).

The weighted moving average is calculated as

$$V_t = m_1 o_{t-1} + m_2 o_{t-2} + m_3 o_{t-3+\dots} m_n o_{t-n} \qquad Eq.2.4$$

Where V_t = Forecast value for the next period, n = Periodic intervals under study

m_i= Assigned weight to the actual observed value for the period t-i .

o_i = Actual observed value for the period t-i

All the weights assigned should sum up to unity.

$$\sum_{i=1}^{n} m_i = 1 \qquad Eq.2.5 \quad \text{Kuo C. et al. (1995).}$$

Contributions of Kuo C. et al.(1995), Gurani, H. et al. (1999), Chen, F. et al. (2000) are worth mentioning in suggesting the various modification for weighted moving average method to obtain higher prediction accuracy. Kuo C. et al. (1995) suggested two methods to assign the weights. One is based on the subject matter expertise regarding the demand of the product and other one is by trial and error. He further reported that the weights are chosen based on any of the two methods or a judicial combination of both. Gurani, H. et al. (1999) argued that as a usual conventional rule, recent past data is the most crucial pointer to the future prediction. Hence, higher weightage is assigned to data which is nearer in the time horizon. Chen, F. et al. (2000) explored how the influence of historical data on the future prediction can be adjusted very well with the weighted moving average method.

2.3.4 Exponential Smoothing

Serious limitation of basic methods of forecasting like simple and weighted moving average is the requirement to recurrently hold a large amount of past data. The older data is replaced by new data and the new forecast is computed. In many applications, the latest happenings are more representative of the future than those which are distant in the time horizon (Frohlich, M.et al. 2002). This premise works out a basis for the other method of forecasting which is called exponential smoothing. Logical reasoning behind this method is that the significance of the data in forecasting practice weakens as the past becomes more distant.

Under exponential smoothing method, following formula is used for calculating the forecast.

New predicted value = Earlier predicted value + a fraction of the prediction error.

New prediction = Earlier prediction + α (Most recent observation – Earlier prediction)

where α (alpha) is known as the smoothing constant.

Or mathematically,

$V_t = V_{t-1} + \alpha (o_{t-1} - V_{t-1})$

i.e $V_t = \alpha o_{t-1} + (1- \alpha) V_{t-1}$ Eq. 2.6 (Heikkila, J. 2003)

where

V_t = Forecast obtained by exponential smoothening for period t.

V_{t-1} = Forecast obtained by exponential smoothening for the previous period.

o_{t-1} = Previous period actual demand.

α = Smoothening constant.

The works of Ryan et al. (2000), Lee H. L. et al. (2003), Zhao X. et al. (2002) have used exponential smoothening as the demand forecasting approach and investigated into different factors which decide the achievable prediction accuracy. Ryan et al. (2000) concluded that exponential models have remarkably higher accuracy in most of demand forecast situations. They have relatively easier computational complexity. When used in computerized demand management system, storage requirement are minor because of the very less use of past data history. In the exponential smoothing method, the data required for forecast is very limited, only three pieces of information are needed to predict the future demand: the most current prediction, the observed demand during that forecast period and a **smoothing factor alpha (α).** The degree of smoothening is governed by smoothening constant. It also determines how fast the forecast system responds to the variances between predicted value and observed value.

Zhao, X. et al. (2002) reported that more recent data assumes higher significance and greater influence in the future forecast. He revealed that main advantage of this method is that it is an adaptive forecasting system. Due to its adapting nature, the system adjusts or modifies itself continuously when the data set is renewed to incorporate more recent data. Hence, exponential smoothing forecasting technique has become an integral part of most of ERP system.

Lee H. L. et al. (2003) conclusively argued that the selection of smoothing constant α is crucial for accurate forecasting. It is done by trial and error method with the expert knowledge of the research analyst in the appropriate domain field of demand management. Smoothing constant is adjusted so that forecasting error reduced to minimum over a time horizon. It has been observed that values in the range 0.1 to 0.3 offer a suitable basis to start with.

2.4 **ARTIFICIAL INTELLIGENCE APPLICATIONS IN DEMAND FORECAST**

2.4.1 Limitations of Quantitative Methods of Demand forecast.

Although the quantitative methods referred to in the previous sections function well, they suffer from some serious drawbacks. Garetti, M. et al. (2000), Zhao et al.(2002), Chandra, C. et al.(2005), Tugba Effendigil et al. (2009) revealed limitations of quantitative methods of forecast. Garetti, M. et al. (2000) established that lack of sufficient proficiency in the respective domain might cause a deficiency in the proper mapping of relationship between the independent and dependent variables, causing an inferior regression. Another important point is that an enormous quantity of data is often necessary to be assured of a precise forecast.

Zhao et al. (2002) used the advanced versions of the conventional methods. Main tools which need to be mentioned under this category are time series models such as complex Box-Jenkins method, moving-average and exponential smoothing, Causal models, like econometric models and regression have also been explored. He concluded that with non-linear data offering a great challenge to be mapped exactly, meaningful forecast is difficult to be obtained even with these advanced versions.

The influence of forecasting approaches on the successful functioning of supply chain was analysed by Chandra, C. et al.(2005) using a computer simulation model with one capacitated supplier and number of retailers in the uncertain demand scenario. He analysed that data which are outside the normal range can cause erroneous evaluation of the standard parameters with all the quantitative technique.

Tugba Effendigil et al. (2009) proposed that some of these drawbacks of quantitative demand techniques can be improved upon by the application of different AI techniques like neural networks. It has been mathematically proved that neural networks have the capability to approximate any functions without being influenced by the above listed limitations.

2.4.2 **Advantages of AI techniques**

AI technique also referred to as soft computing is a group of unique approaches to the general problem solution using human like thinking [Dorffner, G., 1996, Vakharia, A.J. 2002, Al-Saba et al. 2007, Mark Ko et al. 2010]. Some of main the methodologies which come under this category are Evolutionary Algorithms (EA), Neural Networks (NN), Fuzzy Logic (FL) and Expert System (ES). These techniques furnish adaptable information processing and data handling competencies to unravel real-life complicated scenario. The advantages of employing AI technique is its ability to endure inexactitude, ambiguity, and fractional truth to accomplish manipulability and reliability on mimicking human decision-making competence [Mark Ko et al. 2010, Pal, S. et al. 2004, Roy, R., et al. 1999, Tettamanzi, A. et al. 2001].

Soft computing forecasting practices have been receiving ample responsiveness recently in problem solving applications where traditional methods have failed to yield good results. It has been demonstrated that AI techniques have the capability of human learning, by accruing information, acquaintance and familiarity through recurring learning actions [Tugba Efendigil et al. 2009, Yager, R. et al. 1994, Chiu, M.,2004]. Various research studies have assessed the potential of AI techniques in comparison with traditional methods such as regression and moving averages in the field of demand prediction. The studies have concluded that conventional approaches cannot match the capability of AI based systems in terms of accuracy of forecast results [Hung, J.C., 2009, Park, J.I. et al. 2010, Silva, C.A., et al. 2005].

2.4.3 ANN in demand forecast

Intellectual thought process, reasoning and learning process of human brain is attempted to be replicated through Artificial Neural Network (ANN) models. Substantial success in this attempt has promoted the application of ANN in different business solutions. Demand forecast is a prime candidate for ANN application. Demand forecast can tap the efficient modelling capability of ANN for inadequately understood problems for which ample data are available (Wong, B. K. et al. 2007). The capability of ANN to learn by examples is very well exploited in demand forecast and the neural network is trained with the records of a past response or historical data (Wei et al.1997).

With the revolution in AI techniques, neural network has become one of the primary approaches for demand prediction in supply chain management. Specifically, capability of Multi-layer feed forward neural network in modelling any linear as well as nonlinear and function accurately for demand forecasting has been well examined [Aburto, L. et al. 2003 , Faraway, J. et al. 2008].

Preliminary application of artificial neural networks (ANN) started with the energy utility companies to predict short or long term demands for electric load (Al-Saba et al. 1999, Beccali et al. 2004). Wang, T. et al. (2006), Funahashi, K. (2009), Cybenko,G (1989), Singh, P et al. (2007) established through their investigation related to research work that ANN is a universal function approximator and that its capability to map any linear or nonlinear domain is better than various traditional methods.

De Carvalho et al. (1998), Hill et al. (2006), Luxhoj et al. (2006) , Aburto et al. (2007) , Lau et al. (2013), Nikolaos Kourentzes (2013) Lolli F. et al. (2017) have rendered immense contribution in the field of application of neural network for demand forecast. Considerable amount of work was carried out by De Carvalho et al. (1998) in matching and blending traditional and neural network based forecasting methodology which suggested that neural network can improve the performance of prediction. Hansen et al. (2003) examined the conceivable causes for inferior forecast of traditional methods and inferred that use of neural networks is the best solution. Hill et al. (2006) analyzed demand forecasting problems employing this artificial intelligent technique and concluded that neural network can function considerably well in forecasting problems.

Luxhoj et al. (2006) worked on a hybrid econometric neural network model through which he proved that forecasting accuracy of total monthly sales of a Danish company can be improved considerably. This model portrayed the integration of the essential trait of non-linear pattern recognition features of neural network with the econometric models. Aburto et al. (2007) proposed a stock renewal system for Chilean supermarket which works on the basis of a hybrid intelligent system with the fusion of autoregressive integrated moving average models and neural network for predicting the demand in supply chain.

Lau et al. (2013) have presented an exact methodology, the minimum description length (MDL) to decide on the best artificial neural network (ANN which can improve the forecasting accuracy). Their research work proved that balancing methodology of the surrogate data method and neural network creates a complete and detailed structural background for making various demand forecast which can be applied to an extensive range of practical data.

Nikolaos Kourentzes (2013) proposed a neural network methodology to predict the intermittent demand. These NN were used to predict the dynamic demand rate forecasts which did not assume constant demand rates. Lolli F. et al. (2017) used single-hidden layer neural network architecture trained by back-propagation to predict the intermittent demand and studied the application of extreme learning machines algorithm because of their lower computational complexity and good generalisation ability.

2.4.4 **Application of genetic algorithm and hybrid models for demand forecast**

Genetic Algorithm (GA) is used as a tool to evaluate the forecasting model parameters as in Chiraphadhanakul et al. (1997), Jeong B et al. (2002). Numerous applications of GA can also be noticed in many research works as a component of fusion algorithms with other heuristics such as simulated annealing neural networks, taboo search and application- specific heuristics [Kim D. et al. 2009, Ju, Y. K. et al. 1997].

Escoda et al. (1997), Du and Wolfe (1997), Kuo et al. (1998), Kuo et al. (2002), R. Carbonneau et al. (2008), Jamal Shahrabi et al. (2013), Komgrit Leksakul et al. (2015) had conducted research on application of GA and Hybrid algorithm for demand forecast.

Du and Wolfe (1997) made a detailed analysis of application and implementation details of different AI techniques in varied business functionalities like inventory and quality control, planning and scheduling, application of group technology and also forecasting. Escoda et al. (1997) concentrated on the deployment and enhancement of linguistic variables using ANN and Fuzzy Neural Network in the study on product demand. Kuo et al. (1998) presented an intelligent sales forecasting system which worked on the basis both quantitative and qualitative inputs and by the combined power of ANN and FNN. Kuo et al. (2002) examined the hybrid application of neural network and other AI techniques in demand forecasting in a scenario of uncertain customer demands. He modeled the use of fuzzy inference system with adaptive network and artificial neural networks to handle fuzzy demand with inadequate system data. His target area was demand forecast for multi-level supply chain structure.

R. Carbonneau et al (2008) quantified the bullwhip effect by predicting the distorted demand at the end of supply chain using the application of the advanced machine learning techniques, including support vector machines, neural networks, recurrent neural networks. He made a comparative study of these methods with conventional techniques like moving average, linear regression and naive forecasting.

Jamal Shahrabi et al. (2013) developed a new hybrid intelligent model by the integration of genetic fuzzy expert systems and data preprocessing for improving the demand forecasting accuracy in the tourism industry. The new model was named the Modular Genetic-Fuzzy Forecasting System (MGFFS). The accuracy of demand forecast by this new model based on the MAPE and RMSE evaluation was found to be far better when compared with conventional time series models, neuro fuzzy models.

Komgrit Leksakul et al. (2015) suggested an organized and logical method for off-season longan fruit supply forecasting in Thailand using various machine learning tools. He established the supremacy of Fuzzy Support Vector Regression (FSVR) in accuracy of demand forecast over other machine learning like neural network, fuzzy neural network.

In spite of such a vast literature for demand forecast using AI technique, it is found that study on demand forecast of semi make to order industries like industrial valves is rare. Even if neural network is used for many prediction analysis, different network architecture like Radial basis neural network have not been explored to improve the prediction accuracy.

2.5 INVENTORY MANAGEMENT

2.5.1 Inventory - An Introduction

Inventory represents goods or materials that are owned by an organization for future usage. Such goods include (i) raw materials, (ii) purchased parts, (iii) components, (iv) sub-assemblies, (v) work-in-process, (vi) finished goods and (vii) supplies [Harris, F. W. 1913, Burgin, T. A. et al. 1967, Yi Tao Loo et al. 2017].

One main reason that company stores sufficient inventory is that it is seldom possible to exactly forecast / predict sale levels, production times, demand and usage needs. An inventory system is the group of policies and controls that monitors levels of inventory, in order to minimise the inventory cost and to guarantee a smooth operation of the organization [Phillips,D. et al. 1976, Yazgi Tu et al. 2008, Garcia et al. 2013].

Inventory control ensures a balanced bargain by establishing a judicious trade-off between carrying and obsolescence costs of excess stock on one hand and lower service level and lost sales cost due to too little stock on other hand. Inventory management would ensure an optimum service level without maintaining unreasonably excess inventory that are expensive and demanding to handle (George Nenes et al. 2010).

One of the important purposes of managing stock is to resolve these potentially contradicting goals [Zoller, K.1997, Davood Mohammaditabar et al. 2010, Hindriyanto et al. 2012]. The main purpose of Inventory management is to identify the inventory level which would reconcile these potentially diverging goals.

- Optimising customer satisfaction.
- Increasing efficiency of production and purchase functions.
- Optimising financial outlay in inventory.
- Maximizing profit, return on inventory and return on asset.

Inventory serves as a cushion against indeterminate and unstable demand / consumption and keeps a steady supply of items available until replenishments are received. Therefore an inventory system essentially answers two important questions (Leenders, M. et al. 1985).

- What should be the optimum re-order size?
- When should the reordering to be done?

Fig 2.4 represents a basic inventoy replenishment model which graphically explains the the relevant technical terms like reorder point, lot size, safety stock, lead time etc.

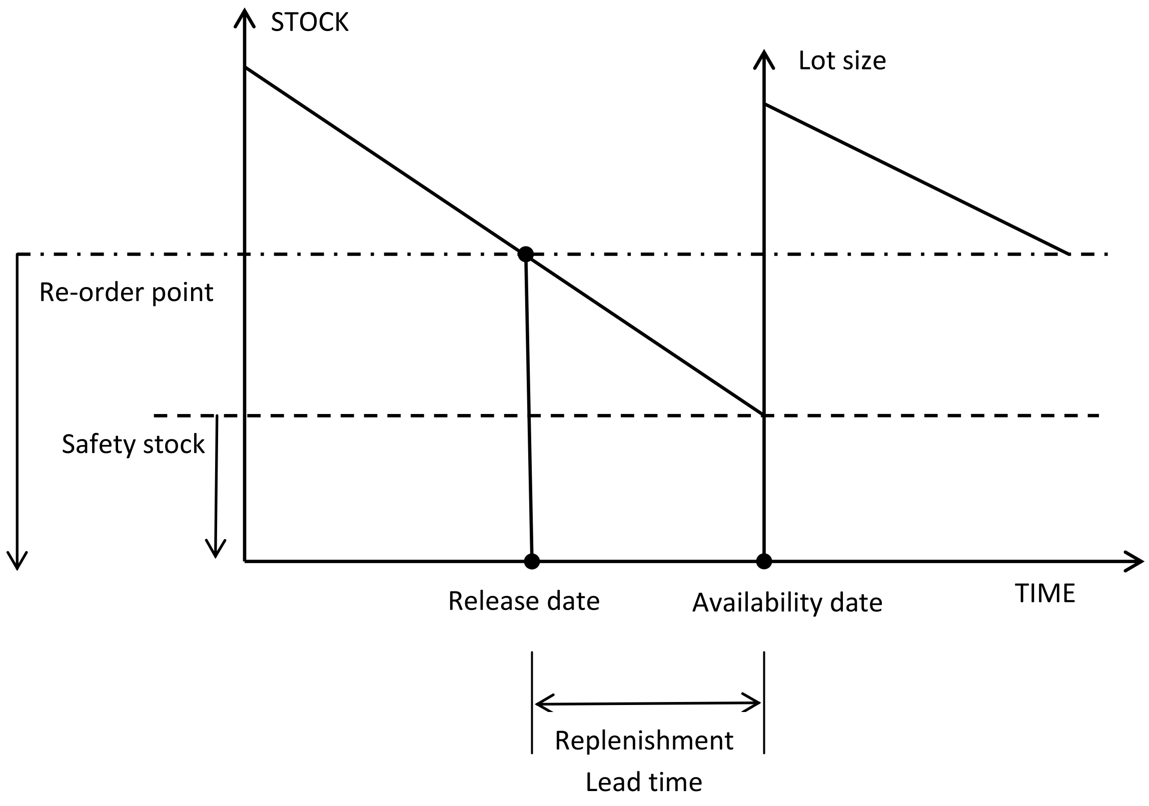


Fig 2.3 Inventory Management Basics (Courtesy: Krone,L.,1964)

2.5.2 Inventory Costs

Decision on optimum inventory level requires careful consideration of the following costs [Silver, E. A. et al. 1990, Samak-Kulkarnia et al. 2013, Hira, D. S. et al. 2009].

2.5.2.1 Holding or carrying costs

This broad classification comprises of the costs for storage facilities, handling and insurance. The cost of pilferage, breakage and obsolescence are also included under this heading. Costs due to depreciation and taxes are also components of holding cost. The opportunity cost of capital is another important constituent. Evidently, larger holding costs tend to support lower stock levels and frequent renewals or top ups [Silver, E. A. et al. 1990, Alejandro Serran et al. 2017, Torkul,O. et al. 2016].

2.5.2.1 Setup or production change costs

This cost is related to production preparation like the tool set up, obtaining or shifting certain material, arranging specific equipment set up, clearing the previous material stock.

Main objective of efficient production planning and inventory management would be to reduce the set up cost. Low set up cost would encourage and justify smaller production lots. This results in low inventory levels and subsequent saving in cost. Just In Time(JIT) production system or lean production faces the challenge of reducing the set up cost to minimum to allow smaller lot sizes [Tamas Koltai et al. 2009, Torkul,O. et al. 2016, Minghui Lai et al. 2016].

2.5.2.3 Ordering costs

This is the administrative cost involved in ordering the material. It includes managerial and clerical cost to prepare the purchase or production order. Ordering cost includes the cost incurred to the company to maintain the system which is used to prepare the purchase order and to track the orders or carry out the follow up activity [Samak-Kulkarnia et al. 2013, Longsheng Cheng et al. 2016].

2.5.2.4 Shortage costs

When the inventory of an item is exhausted or in a stock out scenario, either of two things can happen. The customer will wait for the order to be executed when the inventory is replenished. Effectively a back order is created and filled at a later date depending on the customers approval. Another possibility is that the order gets cancelled which is referred as lost sale. Striking a balance between higher carrying stock to satisfy the demand on one side and costs from lost sales and back orders on the other side is a difficult proposition. This is because of the difficulty in evaluating the lost profits, effects of lost customer base or agreed terms of lateness penalty. Shortage cost is very difficult to determine as it is more of a notional cost [Fiestras-Janeiro et al. 2013, Azzi, A. et al. 2014, Dilay Celabi, 2015].

2.5.2.5 Cost of the item

Discounts and price breaks are important influencing factors under the cost of the item which decides lot size for procurement.

Combined effects of five individual cost components have to be considered to determine the optimal lot sizing either for internal or external procurement. These are holding costs, setup costs, ordering costs, and shortage costs and cost of the item itself. When to order is also a critical factor along with how much to order which will influence inventory cost [Christopher, M., 1992, George Nenes et al. 2010, Dilay Celabi, 2015].

2.5.3. Inventory models

An inventory system can be modelled quantitatively based on demand patterns [Bretthauer, K. et al.1994, Shu-Chin Chang et al. 2016]. They are

- Deterministic inventory models in which demand rate of an item is assumed to be constant [Alejandro Serran et al. 2017, Jui-Jung Liao et al. 2013].
- Probabilistic inventory models where the demand for an item fluctuates and is specified in probabilistic terms [JiSun Shinn et al. 2015, Biswajit Sarkar et al.2013].

Based on the frequency at which orders are placed for procuring inventory, there are two models. They are single period and multi-period inventory systems.

2.5.3.1 Single Period models

Typically orders are made only once. They are also known as the Dollar Limit System and are used for one time ordering for seasonal products or spare parts purchases. Classical example is the newspaper vendor problem [Zhong Yao et al. 2011, Baruch Keren,2009, Chia-Shin Chung et al. 2013]. It models the trade off scenario of optimizing the number of papers to be stoked on the stand. Potential conflict is between the condition on one hand where too many papers are stocked resulting in loss due to unsold newspapers and other scenario where too few papers are stocked and there will be opportunity loss due to lost sales. Single-period inventory models are adopted gainfully in a wide variety of service and manufacturing applications like overbooking of airline flights, ordering of fashions items or any type of one time order.

This is a popular inventory model with the non deterministic demand. Useful life of the product is considered to be very short or only one planning cycle. (Hadley et al. 1963), (Khouja,M.,1999), Hsu et al.(2008) and Chung et al.(2011) developed replenishment policies for products with short life-cycle under different demand conditions and constraints situations.

2.5.3.2 Multi Period models

In the Multi period inventory models, orders are placed multiple times over the entire production cycle. Multi period inventory systems are planned to guarantee that the component will be in the inventory continuously without any period of non availability. Usually the component may be ordered number of times during the planning horizon where the program in the system decides on the lot size of ordering and the timing of the order. Based on the pattern of reviewing current inventory, they are further classified into [Leenders et al. 1985, Ilkay Saracoglu et al. 2014, Leopoldo Eduardo et al. 2014].

i. Continuous Review (also called Fixed Quantity or Q system), where Inventory is reviewed continuously and when inventory drops to a certain prefixed reorder level, a fixed quantity is ordered. This model is generally used for high volume, valuable, or important items [Ilkay Saracoglu et al. 2014, Manuel Cardos et al. 2011, Mahdi Tajbakhsh, M. 2010].

ii. Periodic Review (also called as P system), where inventory is reviewed at prefixed periodic intervals irrespective of the levels to which inventory drops and an order is placed to bring up the inventory to the maximum level. This is used for moderate volume items [Manuel Cardos et al. 2011, Yi Tao et al. 2017, Chiang, C. et al. 1999].

The fundamental difference between these two models is that fixed–order quantity models are "event activated" whereas fixed–time period models are "time activated."

Table 2.1 Comparative study on basic features: P & Q models (Leenders et al. 1985).

Features	Q-MODEL Fixed–Order Quantity Model	P-MODEL Fixed–Time Period Model
Quantity ordered	*Q*—constant (the lot size is same or quantity ordered each time is same)	*q*—variable (Order quantity varies each time dependent on the demand)
Time of placing the order	*R*—when the stock level reaches the reorder point	*T*—In the beginning of scheduled time interval known as review period.
Requirement for maintaining the document.	Each time a stock amendment is made.	Updated only at review period
Inventory size	Smaller than P-model	Greater than Q-model
Time and resources required to maintain	Greater due to requirement for continuous maintaining of records	

That is, with a fixed–order quantity model an order is triggered when the stock level drops to a particular reorder level. This event may transpire at any time, depending on the demand for the items considered. In contrast, in the fixed–time period model, orders are placed at the end of a scheduled time period; only the time is the trigger for order initiation in this model. Table 2.1 shows the comparative study on basic features of P & Q models.

2.5.4 ABC analysis

For companies that keep inventory of large number of items, it is impracticable to give same attention to each item. It becomes imperative for managers to segregate these items based on their relative importance rating so that each inventory class can be suitably controlled [Cohen et al. 1988, Wan Lung Ng, 2007]. ABC analysis is a mostly used efficient method to categorise stock components into particular groups that can be managed and controlled independently.

Traditional ABC analysis sorts out inventory items into three types: A, B or C on the basis of yearly consumption value of an inventory component. This method was first devised by General Electric in 1950s [Guvenir, H.A., 1998, Min-ChunYu, 2011]. The segregation plan is centered around the Pareto principle, or the 80/20 rule, that utilizes the thumb rule- 'vital few and trivial many', which means 20 percent of highly important items and 80 percent less important items. Yearly usage value is calculated by multiplying the price of each item by the annual consumption rate. Flores et al. (1987), Cohen et al. (1988), Partovi et al. (2002), Ramanathan (2006) and Jamshidi et al. (2008) contributed significantly through their research in this field of study.

In the study by Flores et al. (1987), inventory items were organized in the descending order of their annual dollar usage. Class A items are comparatively smaller in quantity, but higher in yearly consumption value. In contrast, class C items are relatively more in number, but account for a smaller amount of yearly consumption. Class B items are those which fall in between class A and C. In the study by Cohen et al. (1988), the application of ABC analysis has been extended by the use of multi criteria inventory classification. Their research works focused on factors other than annual dollar usage. Some of these factors were lead time, product durability &obsolescence. Ramanathan et al. (2006) focused on inventory cost and order size requirement as the basis for classification and observed that inventory planning was more efficient with these methods.

Jamshidi et al. (2008) employed the analytic hierarchy process (AHP) for classifying inventory into A, B, C categories in order to incorporate the use of both quantitative and qualitative classification criteria.

2.5.5 Advanced inventory models

Economic order quantity (EOQ) is one of the extensively used models to address various productions and inventory management challenges. This is a basic model which was proposed by Whitney as early as 1966. In its earliest form, it can be applied for planning a single item in single period with many assumptions. Adopting these assumptions has the advantage of simplifying the model. The model becomes so simple that it will be far away from the complexities of real world situations, thereby restricting its applicability. Literature on inventory management is full of research work dissertations on extensions and modifications of EOQ model and periodic review inventory models which are aimed at reflecting real life scenarios. They extended various approaches for different complicated inventory models like inventory models with finite replenishments, with shortages, with price breaks and discounts, by considering the time value of money using different inflation rates, single item and multi item inventory models etc.

Works of Dunsmuir et al. (1989), Benton, W. C. (1991), Das K.et al. (2000), Chang et al. (2001), Syntetos et al. (2006), George Nenes et al. (2010), Leopoldo Eduardo et al. (2012) etc. suggested different advanced inventory models for the cost optimization and service level maximization for the customers. Dunsmuir et al. (1989) worked on advanced EOQ model to integrate different practical situations like assessments on quantity discount under variable conditions of multiple suppliers, multiple items and resource limitations. He mathematically simulated the interaction between fixed customer service level and a continuous review inventory system for defining reorder levels to maintain a required customer service level. Benton, W. C. (1991) studied the case of highly irregular demand and tried to model the relationship between demand forecasting and continuous review reordering subsystems in the condition of continuous varying demand.

Das K.et al. (2000) worked on an inventory model for multi items in the constant demand and infinite replenishment scenario with the limitations on storage area and total average inventory investment cost. Additional constraint of total average shortage cost was also studied.

Chang et al. (2001) explored the use of a linear programming model created around piece-wise linearization techniques to compute the lot size and reorder point in the situations of variable lead-time and crashing cost. Price and quantity discount were also built into the model. Syntetos et al. (2006) examined the use of different forecasting methods in relation with a periodic review order-up-to inventory control policy to address the intermittent demand issue. The unique characteristic of their model is the application of a gamma distribution with a peak at zero to portray the no demand period frequency distribution. This special case of intermittent and fast moving demand is well integrated into this model with the combination of gamma density and probability mass at zero. George Nenes et al. (2010) carried out a case study on the inventory management in the irregular demand scenario and suggested an effective procedure for precise computation of the base stock levels. His focus was on the study of periodic review system based on gamma distribution.

Research work of Leopoldo Eduardo et al. (2012) focused on vendor managed inventory control system. Multi items were considered under different constraints and classical economic order quantity model was extended to suggest a simple heuristic algorithm. It was proved that this algorithm works better than all other previous models on both the evaluation parameters of total cost and execution time for computation.

2.5.6 Multi Period Periodic Review Inventory Models

Decision on the procurement of multiple items over multiple time periods will be based on the best optimization consideration of different cost objectives to establish ideal lot size and timing of purchase within the procurement horizon. Different costs which have to be considered to arrive at the best tradeoff are purchasing cost, ordering cost, transportation cost and inventory carrying cost. Shortage cost is also another important parameter which needs to be given due importance [Devendra Choudhary et al. 2011 Behnam Vahdani et al. 2017]. It makes business sense for the supplier to offer discounts when larger quantities are ordered. The company can exploit this opportunity to reduce the purchase cost and also the ordering cost if it can increase the lot size of procurement provided the larger holding cost incurred should be balanced by the savings on purchase and ordering cost. The operating advantage of economy of scale is additional benefit of having larger lot size. In such situations, the items could be carried forward for the consumption during next planning cycle. But there would be additional inventory carrying cost.

The consumption for the planning period is met either by the material procured during that period or by the material carried forward from the last planning period. If procurement is done through smaller lot sizes, the carrying cost will be reduced, but ordering cost would increase. On the other hand, larger lot size procurement strategy would reduce the ordering and transportation cost, but would increase the inventory holding cost substantially. Supply chain disruptions resulting in late deliveries and quality rejections would also influence the purchase lot sizing decisions. On the whole, lot sizing decisions would be driven by the trade off between purchase cost, ordering cost on one hand and carrying cost on the other hand over the total procurement horizon [Robinson et al. 2009, Ann M. Noblesse et al. 2014].

Wagner et al. (1958), Aggarwal et al. (1993), Pratsini (2000), Brahimi et al.(2006), Smith et al. (2009), Devendra Choudhary et al. (2011) and Woon-Seek et al. (2015) have done significant contributions in this subject of multi period periodic inventory model. Exact solution for multi period single product inventory lot sizing was first presented by Wagner et al. (1958). The concept of Dynamic programming was used in obtaining the solution. Aggarwal et al. (1993) further attempted through their research work for the improvement of the optimum solution suggested by dynamic Programming. The solution consisted more realistic input constraints to reflect real world scenario.

Brahimi et al. (2006) reviewed the single item lot-sizing problem considering both uncapacitated and capacitated versions. He presented his work involving the study of a number of significant extensions to classical lot-sizing models, altering its basic characteristics like planning horizon, number of levels, number of products, capacity or resource constraints, deterioration of items. Joint procurement and production decision problem for single item, multiple period time horizon was studied by Smith et al. (2009). The objective function maximized the profit under the capacity and inventory constraints. They considered decision variables, such as sales price, production quantity, and sales amount for a single item.

Rejections, late deliveries and quantity discounts for a multi-period procurement lot-sizing problem for single product and single supplier were modeled by Devendra Choudhary et al. 2011. Integer linear programming methodology was employed by him to obtain the solution where he tried to optimize cost objectives by procuring the material in appropriate lot size and at appropriate time.

A goal programming model was suggested by Pratsini (2012) taking into account price, quality and delivery objectives to plan procurement for single product over a defined scheduling horizon. Single level, multi item with capacity constraint inventory model was studied by considering the effect of set up learning. He elaborated upon the lot sizing model for this problem by developing a heuristic to examine the consequences of set up learning on production planning.

Woon-Seek et al. (2015) suggested an inventory model to study the simultaneous impact of the procurement quantity and the shipping policy on the aggregate costs, which comprises of components on production cost, inventory carrying cost, and consignment cost. The model determined the optimum lot size and transport policy so that total cost is minimsed. A heuristic algorithm with a modification mechanism was suggested based on the ideal solution properties.

2.5.7 AI application in Inventory management

Over the past two decades, AI technology has emerged as an important development in the field of information science. A number of artificial intelligence techniques including fuzzy logic and genetic algorithms, neural network, ant colony optimization, particle swarm optimization have been deployed to enhance effectiveness and efficiency in various aspects of inventory management [Mark et al. 2010, Angappa Gunasekaran et al. 2014, Borja Pontet et al. 2017]. Inventory management decisions have to be worked out in uncertain demand environment. They are also characterized by optimization of multiple objectives, most of the times, conflicting objectives like cost and service level. AI techniques provide a valuable tool to address these issues of uncertainty and multi objective optimization (He-Yau Kanga et al. 2010).

2.5.7.1 Application of Fuzzy Logic in Inventory management

Fuzzy set theory was first suggested by Professor Lotfi Zadeh during 1960s for mathematical representation of ambiguity, vagueness, imprecision and uncertainty (Zadeh, L. A., 1965). The use of approximate information and uncertainty in decision making by human reasoning is modelled though Fuzzy theory. It has further been used to create validated tools to take care of the imprecision and vagueness which is inherent characteristic of a number of real life situations [Zadeh, L.A.1978, Giachetti, R.E. et al. 1997, Dubois, D. et al.1986].

Researchers in inventory management field have always found application of Fuzzy theory very fascinating and fertile (Du et al. 1997). This has become even more relevant now because of growing importance of supply chain and logistics management in today's global environment which is characterized by uncertainty and impreciseness (Mark et al. 2002). Ample inventory models have been conceived which uses the Fuzzy set theory to simulate the factors that involve uncertainty, vagueness and ambiguity. Some of the important models were by suggested by the research works of Juite Wanga et al. (2004), Wang Xiaobin et al. (2004), Lin Wang et al. (2012).

Juite Wanga et al. (2004) reported that demand quantity is the main uncertain factor in inventory control which justifies the application of fuzzy theory. To model the supply chain uncertainties and to establish the supply chain inventory strategies considering the ambiguity in various parameters, he worked on a fuzzy decision algorithm which would work very well while there is uncertainty in data or even in case of non-availability of past data. The algorithm which was presented, would help to improve the decision on the inventory strategies by analyzing the fuzzy variables of risk, customer service level and inventory investment in supply chain.

Wang Xiaobin et al. (2004) extended the classical economic order quantity models to include the independent fuzzy variables cost of each unit quantity and order cost of each cycle. Holding cost, lead time, penalty cost, storage area other fuzzy parameters were incorporated into his model.

Lin Wang et al. (2012) reworked the continuous review inventory models to incorporate the fuzzy variables of lost sales rate and lead time. With this, he modeled a situation which allowed shortages and variation in lead time. The model also reflects the real world scenario where a certain fraction of demand is back ordered in case of stock out. The partial information of lead time was modeled using minimax distribution free procedure to find the optimum inventory strategy.

2.5.7.2 Application of GA in Inventory Management

Genetic algorithm is search and optimization technique which works based on natural evolution philosophy. Due to their robustness and adaptive nature, an intense interest had been generated among the researchers in the field of inventory management [C.A. Silva et al. 2005, Dilay Celabi, 2015, Maryam Akbari Kaasgari et al. 2017]. Traditional search and optimization methods including exact methods are effective only in the cases where the number of variables to be optimized is limited. But Genetic algorithm, by adopting certain principles of biological evolution, can work very well for multi objective optimization. Research fraternity strongly contends that Genetic algorithm has a lot of unexplored potential for the application to variety of field in Inventory Management [Goldberg et al. 1989, Ali Diabat et al. 2016, Ilkay saracoglu et al. 2014].

Maiti et al. (2006), Arindham Roy et al. (2009), Seyed Hamid et al. (2011), Dilay Çelebi (2015) and Ali Diabat et al. (2015) have applied GA for the problems related to inventory management and reported substantial improvement in the results obtained. In the research work by Maiti et al. (2006), GA has been applied to solve the multi objective inventory model involving order quantity and re-order point problem on two storage inventory scenario. He further extended his model and developed a heuristic based on GA to solve the economic lot size scheduling problem. Arindham Roy et al. (2009) suggested a modified genetic algorithm for multi item multi-buyer joint replenishment problem. Compared to conventional approach, his novel GA was able to obtain better optimization results for production and inventory hybrid model for stock dependent demand integrating learning and inflationary effect. Fuzzy GA was used with changing population size method.

Seyed Hamid et al. (2011) suggested a genetic algorithm model to optimize two-echelon continuous review inventory systems. He worked on tuning the parameters of GA to optimize its performance. An effective stocking policy was developed with the goal of optimizing the total annual inventory investment. The model was made more practical by considering restrictions on budget, the average annual order frequency, expected number of backorders.

Dilay Çelebi (2015) studied the spare part distribution system of a Turkish automotive manufacturer under a unified control system. He developed a case study based on his findings which deals with his proposed GA based inventory model to determine optimum stock level and efficient management of distribution network. In his case study, he addressed two echelon inventory control problem with both combinatorial and sequential behavior. He also extended his model to incorporate a large number of specific properties of supply chain which would make it more practical.

Ali Diabat et al. (2015) studied integrated supply chain problem, with the emphasis on a capacitated multi echelon joint location inventory control scenario. A hybrid GA based heuristic was developed which would identify the location of set of warehouses to optimize the total inventory cost including transportation.

2.5.8 Application of AI in multi period periodic review inventory models

Maiti et al.(2008), Reza Zanjirani Farahania et al. (2008), R.K. Gupta et al. (2009), Taleizadeh et al. (2013), Seyed Mohsen Mousavi et al. (2013), Javad Sadeghi et al.

(2014) have investigated the multi period periodic review inventory management optimsation problems through the application of AI techniques. Maiti et al. (2008) extended the GA model from single item to multi-item inventory control problem. He also considered the effects of two types of price discounts, All Units Discount (AUD) and Incremental Quantity Discounts (IQD). Roulette wheel selection, arithmetic crossover and uniform mutation were other important features of GA proposed by him.

Reza Zanjirani Farahania et al. (2008) presented a mixed-integer linear programming model to represent inventory management and distribution network of a supply chain with three echelons. The bi objective model optimized the two objective functions of cost minimization and minimisaton of sum of back orders and surpluses of all products. Constraints included the delivery lead time and installed capacity. A novel approach of non dominated sorting genetic algorithm was used to solve the problem.

An inventory policy was modeled by R.K. Gupta et al. (2009) with uniform demand rate, unvarying lead time, finite time horizon and a discount triggered by advance payment. It was solved by Real Coded Genetic Algorithm (RCGA). The optimal number of cycles and lot size in each cycle was determined along with optimal profit. RCGA was characterized by ranking selection, new approach to arithmetic cross over and uneven mutation.

Taleizadeh et al. (2013) presented a mixed-integer nonlinear mathematical model to represent a multiproduct multi-constraint inventory control problem. He incorporated stochastic replenishment intervals and discount into this model. The problem was solved using Genetic algorithm.

Seyed Mohsen Mousavi et al. (2013) modeled an optimized multi-item multi-period procurement and inventory control scenario taking into account the effects of discounted cash flow and inflation. Two calibrated meta-heuristic algorithms, GA and simulated annealing was used for its solution. The performance of the algorithm was evaluated in obtaining the optimal lot sizing of the products. The objective function minimized the net present value of total system cost over a procurement planning horizon. The model was made more practical by considering the constraints of storage space, budget, and order quantity.

Javad Sadeghi et al. (2014) worked on vendor managed inventory control. Conflicting objectives of reducing the total supply chain cost and maximizing the reliability was modeled using GA. Order size along with the order frequency of the retailers, travelling distance from vendor to retailors and the number of machines required was optimized.

2.5.9 Application of Ant Colony Optimisation in Dynamic lot sizing

Ant Colony Optimization (ACO) meta heuristic is characterized by biologically optimized searching capability of ant colonies. Jui-Tsung Wonga et al. (2011), Ali Roozbeh Nia et al. (2013) and Loius et al. (2013) have applied ACO for the dynamic lot sizing problem of inventory management. Jui-Tsung Wonga et al. (2011) used ant colony optimization (ACO) to solve stochastic dynamic lot-sizing problem. In this novel approach, ANN was utilized as machine learning platform to study the simulation results. Based on the learning, optimal decision variables were determined using the application of real valued improved ACO algorithm. The results were compared with that obtained from response surface methodology and found to be far better with ACO application. In another study by Ali Roozbeh Nia et al. (2013), advanced ACO model was presented which represented multi item economic order quantity scenario under shortage for fuzzy vendor managed inventory. He compared the results obtained by solving this model with the outcome two other meta-heuristics, GA and Differential Evolution Algorithm (DEA). In addition to normal constraints, contractual agreement between vendor and buyer on the number of pallets required to supply the items is considered. Number of deliveries and quantity of order under fuzzy environment were also incorporated. His sensitivity analysis showed that ACO algorithm was best in terms of CPU time of execution whereas differential evaluation gave the optimum total cost.

Loius et al. (2013) developed an inventory control model with the objective of minimizing the supply chain total cost and products' lead time. Consequently, the model optimized the safety stock and lead time throughout the supply chain. ACO based approach was adopted and bi-objective MAX–MIN ant system was used to solve this model.

However, not much work has been carried out on the front of multi item multi period dynamic lot sizing applications with ACO.

2.6 SUMMARY AND GAP IN KNOWLEDGE

This extensive review of literature in demand forecast and inventory management inferred that demand forecast and inventory management are integral part of supply chain management. Inventory management is an essential constituent of organizational core competencies which need to be given prime importance for the successful standing in market place. Strategies adopted by the companies to improve their bottom line and profit margin should cover accurate demand forecast and optimized inventory management. These are the enablers for improved customer service at the minimum operational cost.

Uncertainty in demand has made accurate demand forecast an important contributing factor for the healthy bottom line of organization. Demand forecast assumes a vital role within supply chain management. A trust worth and unfailing demand management can enhance the superiority and power of organisational strategy. But, the conventional demand forecast technique have been found to be inaccurate which as a result would amplify the bullwhip effect across different stages of supply chain. Other limitations of the conventional methods point to the requirement of large volume of historical data and inability to process nonlinear patterns and bias of data which is not within normal range. Most of these limitations are overcome by the use of neural networks, which have been mathematically validated to be universal approximates of functions. The ability of neural network to learn and generalize from the provided set of past data about the patterns in the problem domain of interest, have made them excellent choices for demand forecast application.

Decisions, regarding when to buy and how much to buy, have been influencing the efficiency and effectiveness of supply chain management. Optimising the conflicting objective of cost minimization, profit and service level maximization in the environment of uncertain procurement lead time product demand has drawn much interest from researchers. Artificial intelligence techniques have revolutionized the research work related to inventory management. Fuzzy theory, GA, ACO and Particle Swarm Optimisation (PSO) find vast application in multi objective optimization.

Detailed analysis of existing literature has shown following gaps in knowledge which has presented an opportunity for further research:

- Even if there are number of independent studies using AI applications on demand forecast and optimum lot sizing for inventory management, very less instances of integrated AI application on demand forecast and inventory management can be seen. The output of neural network demand forecast can be used for lot sizing optimization using GA application, which has been rarely explored.
- Most of the studies on demand forecast have concentrated on the MLP architecture of neural network. Very few attempts have been made for exploring the use of other architectures like Radial basis network which can improve the forecast accuracy.
- Most of the studies, either in demand forecast or lot sizing, have worked on the simulated data. Limited number of applications and model validations can be seen from the real industrial data.
- Very few studies have the background of semi-make to order type products like industrial valve which is the subject of current study.
- On the front of periodic review model of inventory management, research literature is full of single item multi period lot size optimizing application of GA. It is observed that very little work has been carried out on multi item multi period lot sizing.

The current work proposes an in depth study of existing inventory management system of an industrial valve manufacturing and trading company and suggests a novel integrated application of AI technique for demand forecast and multi period multi item periodic review inventory model.

2.7 OBJECTIVES OF THE PRESENT WORK

The objective of this research work is to explore the integrated application of different AI techniques to the problem of demand forecast and inventory management for a company dealing with industrial valves. With reference to the above objective, the inventory management system of an existing company specialized in the field of industrial valves has been analyzed, and following major research goals have been set.

- To apply Artificial Neural Network (ANN) for forecasting the periodic demand rate of the product for the company under study and explore the different architectures to improve prediction performance.
- To apply and validate the ant colony optimization and GA model for multi objective optimization problem of multi product multi item multi period procurement lot sizing, based on ANN results of periodic demand rate.
- To compare the performance of developed ACO and GA model based on the different evaluation parameters.

2.8 SCOPE

Based on the proposed objective of the current study, the scope for the research work in the area of demand forecast and inventory management includes:

- A detailed study of the inventory management system of existing company dealing with industrial valves and current demand forecast techniques which is being followed.
- Collect information regarding the historical demand data which will be used for training the neural network
- Modelling of ANN with MLP architecture for demand forecast and ascertaining the forecast accuracy.
- Exploring the use of different architecture of radial basis neural network for the demand forecast model with the objective of improving the prediction accuracy.

- Usage of the constructed ANN model to forecast the periodic demand rate for the next periods which will be used as the data for the inventory model.
- Identify and gather all the data regarding the various inventory costs and the budget and space constraints which is necessary to formulate the inventory model.
- Modelling of ACO & GA for multi item multi period periodic review inventory lot sizing problem to optimise the ordering, carrying and purchase cost with the budget constraints , space constraints and reserve stock constraints.
- Tune the parameters of GA based on Taguchi design of experiments and evaluate the effectiveness of the model.
- Validate the developed models and compare ACO and GA models based on the performance parameters of minimized cost objective function and CPU time of execution.

Chapter 3.

RESEARCH METHODOLOGY

Based on the literature survey, it is observed that very little work has been done in obtaining optimal solutions for the inventory problems utilizing the combined benefits of neural network and genetic algorithm techniques.

Artificial Neural Network (ANN) is used for forecasting the periodic demand rate of the product for an existing valve manufacturing and trading unit based on the study of its inventory management system. Two different approaches to ANN, namely Multi Layer Perceptron (MLP) and Radial Basis Function Neural Network (RBFNN) were tried out to develop prediction system for demand forecast. To identify the right kind of optimization, two models namely Ant Colony Optimisation (ACO) and Genetic Algorithm (GA) technique were used for multi objective optimization to arrive at the inventory control model based on ANN results.

The data for the inventory management of an existing firm dealing with industrial valves Shalimar valves, MIDC Industrial Area, Ghansoli, Navi Mumbai was used to validate the models for the demand forecasting and multi objective optimization of inventory control system.

3.1 DECISION ON THE RESEARCH APPROACH

The research process consists of various stages like framing and justifying a topic for research, studying the literature, choosing an approach, gathering data, examining, investigating and evaluating data and preparing report and scripting the thesis (Saunders et al., 2003). Saunders et al. (2003) classified the research approach into two ways: Deductive and Inductive. In deductive approach, the researcher develops certain concepts or propositions and a research strategy is designed to authenticate and validate the hypotheses. In the inductive approach, the researcher develops models and concepts based on result of the data analysis after collecting the relevant data.

There are some fundamental differences between deductive and inductive approaches in its application to research process. Deduction approach stresses on the following aspects:

- Developing the theory and then gathering data to validate the theory
- The need to explain causal relationships between variables.
- The collection of quantitative data.
- Legitimacy and soundness of data to be ensured by the exercise of suitable controls.
- The implementation of theoretical knowledge to ensure clarity and intelligibility of definition.
- To ensure that samples are of adequate size in order to generalize inferences.

Induction method focuses the following features:

- Understanding the connotations linked with proceedings and actions.
- A proper and in depth understanding of the research context.
- Gathering quantitative data, and also the information which cannot be expressed in quantitative form.
- Flexibility required in permitting the changes of research direction as the research work progresses.
- Lower importance on generalization.

Based on the research questions and purpose of the thesis, the deductive approach has been chosen as the most appropriate approach to be employed. A wide range of relevant models and principles have been reviewed, new methods have been proposed for demand forecast and multi objective optimization which would help to improve the inventory management of the company under research. The methods have been validated using the data collected from the organization subject to research.

3.2 RESEARCH METHODOLOGY

Fig 3.1 depicts clearly the research methodology adopted. The scope of the research work includes an integrated application of AI tools for the twin objectives of demand

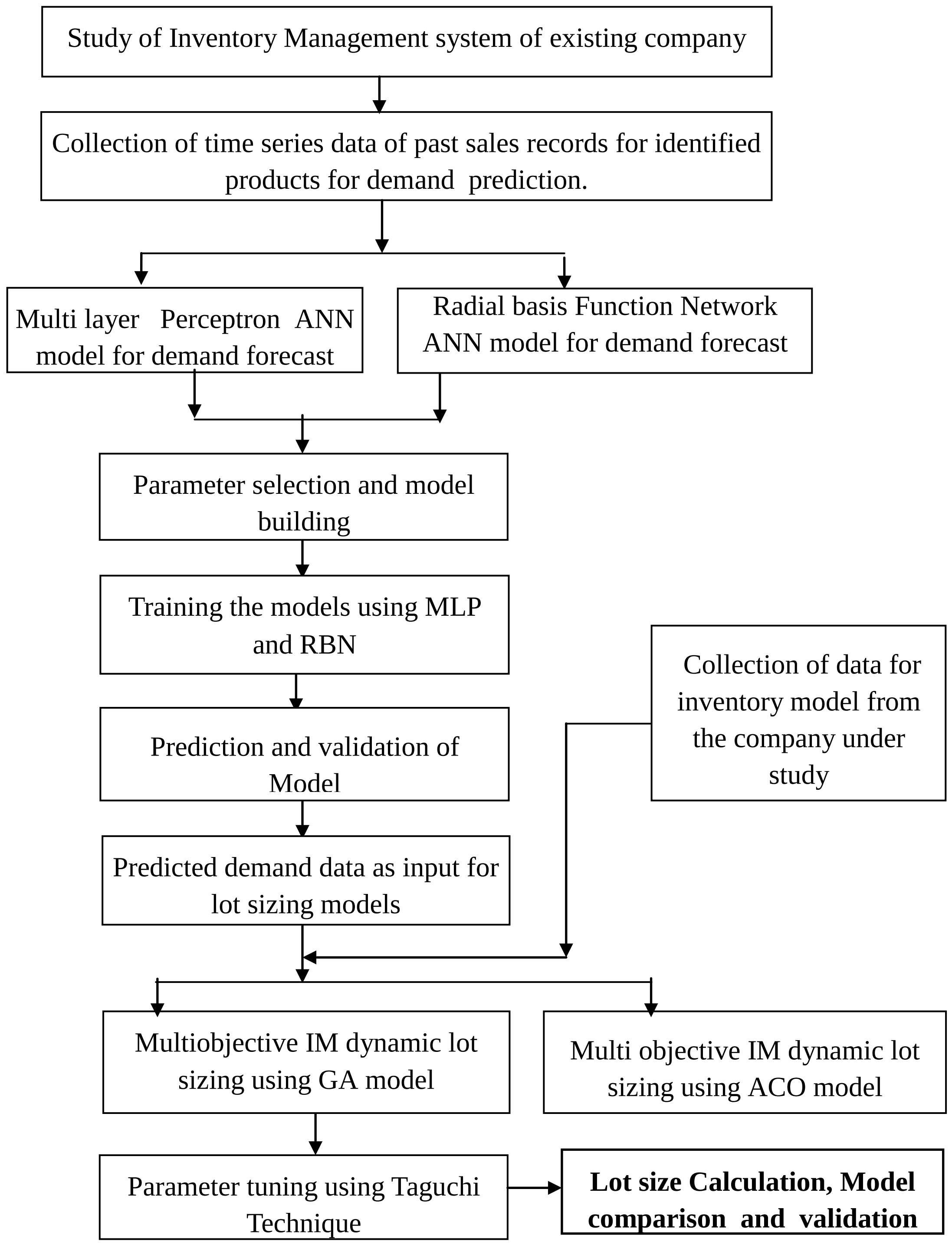

Study of Inventory Management system of existing company
Collection of time series data of past sales records for identified products for demand prediction.
Multi layer Perceptron ANN model for demand forecast
Radial basis Function Network ANN model for demand forecast
Parameter selection and model building
Training the models using MLP and RBN
Collection of data for inventory model from the company under study
Prediction and validation of Model
Predicted demand data as input for lot sizing models
Multiobjective IM dynamic lot sizing using GA model
Multi objective IM dynamic lot sizing using ACO model
Parameter tuning using Taguchi Technique
Lot size Calculation, Model comparison and validation

Figure 3.1 Schematic representation of research Study

forecast and periodic order lot sizing. The inventory management system and practices of company under study is analyzed in depth. Relevant data is collected. Historic sales data for selected product units are collected.

Artificial neural network approach is adopted for the demand forecast. Different network architecture were tried out to improve the prediction accuracy. First, demand forecast was done based on Multi layer perceptron model after training the model with the past sales data of the selected product unit. Prediction accuracy was compared with the demand forecast obtained from Radial basis function architecture neural network.

Based on the best method of forecast, demand was predicted for subsequent periods. This demand data was used as input for the multi objective inventory optimization model to compute the multi period multi item reorder lot size. Conflicting objective of optimizing the order cost, holding cost and purchase cost with different price breaks is modelled using different AI approaches of Ant Colony Optimisation and Genetic algorithm. Optimum reorder quantities, objective function value of minimum total cost of inventory were calculated using different models and compared to identify the best method of optimization. The CPU execution time of algorithm run was also considered as another evaluation parameter.

3.3 REVIEW OF THE CURRENT INVENTORY MANAGEMENT PRACTICES OF COMPANY UNDER STUDY

The company under study is pioneer in the Indian valve manufacturing and trading and through its sister concern has designed and manufactured excellent products for different engineering fields. Since many years, they have indigenously carried out functional research and produced wide range of valves thus earning an international distinction for technical superiority in the field of valve manufacturing and trading. The technical excellence of company has been recognized and they have been awarded with dual ISO 9001, API 6D, API-Q1, API 600 marking. The product range comprises carbon steel valves, stainless steel valves and strainers. Versatility in design and development has given the company an edge over the competitors in its primary capability in developing products that is supplied to a variety of chemical and construction industries.

The company has got a big warehousing facility at Rabale in new Mumbai. Company has got in house manufacturing facility and testing facility. It has got a good marketing network. The company gets most of the valves manufactured through its vendors as per the stringent testing requirements. The company needs a good demand management capability and inventory management lot sizing decision making mechanism for their trading business to reduce total inventory cost.

Acquiring, allocating and controlling the factors of production is important for organisations to achieve its objectives and retain its core competence. For the organization to survive and grow, inventory management is a key activity of business logistics. With this point of view, the organization under study has set one of the key performance indicators to hold inventories at the lowest possible cost, without interrupting the supplies for ongoing operations. Management makes the inventory decisions based on a trade off between the different aspects of cost, such as the holding cost, carrying cost and cost of insufficient inventory. Important points related to inventory management practices by the company under study are highlighted below:

3.3.1 Inventory Planning and Scheduling

This process takes care of determining the stock essential for the organization to ascertain interval of time for smooth functioning of day to day business activity. A good inventory plan is characterized by preparation of stock plan sufficiently ahead of requirement which will facilitate planners to carry out the buying of required material in right quantity at the right period without overstocking and also without causing inconvenience due to quantity and quality deficiency.

3.3.2 Inventory Recording

Company believes that precise and current stores records are important for efficient management of stores. The central business process of inventory management comprises of correct stock taking and prompt maintenance of records of goods receipt or production. Goods issue from the stores is properly approved and the records of approval need to be maintained which can be used as audit trail later. It is responsibility of the stores department to totally eliminate the inaccuracies of stock management and to ensure precise and dependable stock records. Annual and cycle physical inventory checks, surprise and spot checks form a part of monitoring process which ensures an effective inventory management and control. In order to facilitate the auditing and to confirm that each and every transaction is authorized by the competent authority, the company uses many documents. These can be used for legal purpose also. Some of them are listed below:

Purchase requisition note: Document created by either the person responsible for stores or person from user department requesting the purchasing department for procurement of certain material within certain time frame.

Goods received note: Document recording the goods receipt into stores which describes the goods and quantity.

Stock record card/Bin cards: This document for individual material records the goods receipt, issues and the balance in store.

Materials return note: This document records and authorizes the unused inventory to be returned to stores. The goods not used during production or goods which could not be sold are returned back to stores based on this document.

Scrap note: This is a document used for keeping track on scrap produced and its movement across the different departments and stores.

3.3.3 Inventory Valuation

Inventory valuation refers to the process by which the raw material, semi finished goods and finished goods inventory are valuated based on the agreed norms. This valuation is an important input to profit and loss account of the company. There are usually three common methods adopted for inventory valuation: First in First out (FIFO), Last in First out (LIFO) and the average price method. The company adopts FIFO method as it believes that it represents the correct way of measuring the company assets.

First in First out (FIFO) is a valuation technique in which value of material is computed based on the price of the oldest inventory until all the units are used and then the price of second lot is used to determine the value and the process is continued. FIFO method is based on the standard that inventory bought first are issued first. After the first lot or batch of materials procured is consumed, the next lot is prepared for use. The inventory is valuated at the earliest costs.

3.3.4 Inventory control

Inventory control in the organization under study is the activity which organizes the availability of raw material for production and assembly lines and finished goods for sales. It involves the co-ordination of important functions of organization, the procurement, manufacturing, sales and distribution to meet the requirements of marketing division. Inventory control also includes ensuring the availability of not only the current sales items and new products but also consumables, spare parts and other supplies. The company strives through its inventory/stock control techniques to make sure that different kinds of inventory including stocks of raw materials, work in process and finished goods are maintained at appropriate levels which provide highest service level to customers at minimum cost to the company. An efficient Inventory management system should reduce to minimum, the time and carrying costs. It must also be capable of providing required stock for uninterrupted production, sales operation and for satisfactory customer service. Following activates are studied which support inventory control.

Checking Receipts – Quantities of material received in to the stores are checked as per the document either by weighing, counting or measuring. Proper checking of receipts, ensures that the quantities are correct at first instance thereby providing a good basis for further operation.

Checking Issues – Issued quantities and description of material issued are properly checked so that it matches with the goods issue slip, before they are issued for production or sales. The policies and procedure of the company specify that both the issuer and receiver should cross check the quantities and description and sign for it , which would provide a reasonable assurance that there would be no discrepancy between the documented and actual quantities.

Spot checking – In order to have a safe guard against the mal practices related to stores issues and receipts, a practice of random spot cross checks at irregular intervals by the senior stores officers and managers is implemented. This is in addition to regular stock taking. This practice is followed with a good intention of excluding any scope for material misappropriations.

3.3.5 ABC Analysis

The company regards this as the best approach for the inventory management based on the principle of selective control. ABC analysis classifies products on the basis of importance they assume in company operation and financial performance. Importance may be based on cash flows, time required from purchase order stage to goods receipt stage, impact of shortages, sales volume, or profitability. The company gives 80% weightage to annual usage and 20% for the stock out cost. Class A comprises of around 20% of items but constitute around 80% of annual usage. Class B forms around 30% of items by quantity and 15% by value. Rest of 5% by value is class C which makes around 50% by quantity. Company follows the following procedure to classify inventory items into A, B, C categories (Company handbook, Shalimar valves, 2013).

1. Annual consumption of each item is determined.

2. Annual consumption of each item is multiplied by the cost of the item to obtain the total annual usage value.
3. Aggregate annual inventory expenditure is arrived at by adding the total annual usage of all the items.
4. List out all the items and arrange them in the descending value (Annual Value);
5. Accumulate value and add up number of items and calculate percentage on total inventory in value and in number.
6. Percentage of total consumption for each item is calculated by dividing the total annual usage value of each item by the cumulative annual inventory financial outflow.
7. Items are ranked based on the percentage of aggregate consumption.
8. Evaluate annual consumption distribution and categorize items as A, B, or C.

3.3.6 Inventory Levels.

Maintaining the inventory level is a stock management tool, which deals with regulating the amount of inventory stocked by an organization. The main aim of this control is to strike an equilibrium between profitability and liquidity to ensure that there is no shortage or excess of raw materials.

The company has determined the inventory level it needs so that over or under stock is avoided. The company has established the following parameters in order to intelligently escape from adverse stock levels i.e. the re-order level, average stock level and maximum stock level, minimum stock level of safety stock.

Re-order level cautions the stores that the stock has touched the lowest point and it is the time to make stock renewal orders. The re-order level must be adequate enough to make provision for the maximum possible consumption of stock during the reorder period and for delayed deliveries.

Re-order point = Maximum daily consumption x maximum lead time/period (Hadley, G. et al. 1963).

Safety stock or buffer stock is maintained to avoid shortages. It is the stock level of a material below which it should not be permitted to drop. This is an insurance against only unusual situations.

Minimum stock level/safety stock=Re-order level – (average rate of usage x average lead time) (Hadley, G.et al. 1963).

Maximum stock level denotes the highest amount of inventory the company can stock at any time. Maximum level represents the level above which stock should not be permitted to exceed as it becomes uneconomical to hold the inventory due to locked up capital and higher holding cost (Krone,L.,1964).

3.3.7 Two bin system:

The company uses this method for inventory control of relatively inexpensive or non-essential items or items of category C. The inventory is divided and placed in two separate compartments or bins. The items are used from the first bin during normal consumption. New supply is ordered as soon as the first bin is empty. The second bin contains quantity of items that will cover the usage between the dates of placing an order to the date of delivery.

As depicted in Figure 3.2, the two-bin method for re-ordering material can be easily implemented and it offers an easy and direct method for tracking the usage. As in the Part A of pictorial representation, material is used from the first section of the bin only. When all the items are consumed in the first section, the items in second section of the bin are used and an order is placed with the supplier for a replenishment. In the Part B diagram, instead of section, a certain level is used as the reference. When the inventory reaches certain specified level, reorder is placed.

3.3.8 Inventory Costs.

The cost and benefits of the inventory have to be balanced in order to have a successful inventory management.

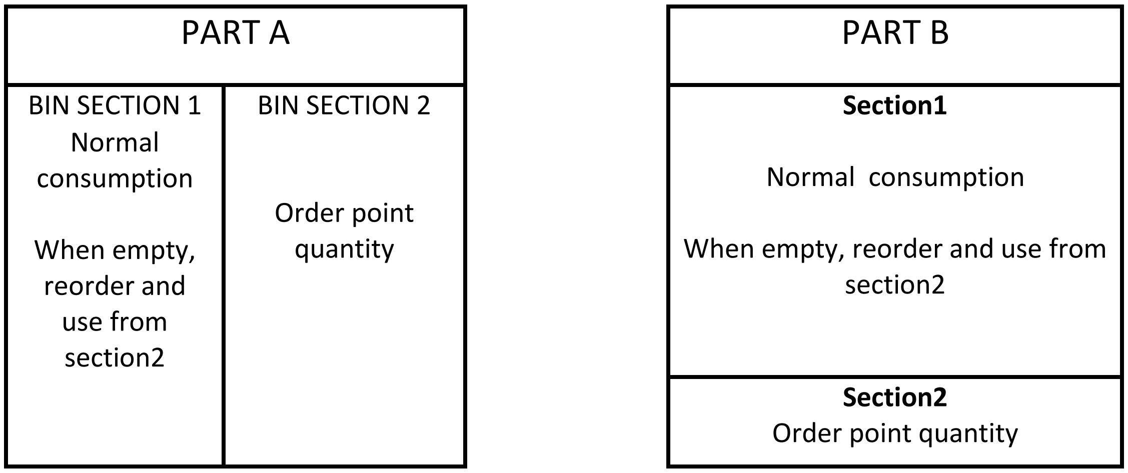


Fig 3.2 Two Bin System (Courtesy: Fei-Long et al.2010)

True cost of inventory, in most of the times, is elusive as it involves not only warehousing cost and cost of capital tied up with inventory, but also the insurance and taxes and cost of obsolescence and stock out. The company has got a very good perspective of this fact.

The main objective of the Inventory management in the company is to ensure that costs associated with inventory are minimized. These cost include holding /carrying costs, ordering costs and purchase costs which make a sum of total stock costs.

3.3.9 Challenges faced by the researched company in the field of inventory management

The company maintains all the store records in the excel format and updates the inventory in the system with the goods receipt and goods issue. The company tried to implement ERP to stream line purchasing and inventory management functions.

Accurate demand forecast being the vital component for a prudent inventory management decisions, the company want to improve their forecast accuracy. At present 3 months moving average method is used for prediction.

Company adopts ABC classification for identifying the material of higher importance. Class A and B items inventory management is done on a continuous basis with the reorder point and maximum stock level. Company finds it extremely difficult to monitor the stock and place the order and monitor it. Because of this the company misses the reorder points leading to short fall of the critical items resulting in delayed deliveries and compromise in service level. There is excess stock for certain items. The company wants to bring down the inventory value, without compromising customer service level. Inventory stock-out often occurs despite the heavy investment in inventory.

The company prefers to have a periodic review and place the order at regular interval. A dynamic lot sizing procedure to be followed with budget and space constraints which optimizes ordering, carrying and purchase cost. With this, the company hopes to minimize the frequent ordering cycles and reduce the missed order and thereby improving inventory management.

Based on the requirements of the company under study, the aim of this research is to apply AI technique to demand forecast so as to achieve higher accuracy of prediction. Further application of AI approach was tried to arrive at the optimum dynamic lot sizing for the periodic reorders based on the forecasted demand.

3.4 DATA COLLECTION METHODS

Primary data was collected mainly through detailed interview with the operations head of company and through observations into the daily activities of stores, goods receipt, goods issue and other operational process of warehouse activities. Data related to the purchase was collected by interviewing the purchase manager and access into the information system related to purchase and material department. Wherever face to face interview could not be held due to logistic difficulties, e-mails were used for data collection and compilation. The interview questions were prepared in advance keeping in mind the exact requirement for the proposed research work. The unstructured interview approach helped in giving better flexibility in data collection so that a very clear picture of facts emerged after the deliberations. The semi-structured interview approach ensured that focus was not diverted away from important identified questions, but at the same time, allowed a certain degree of flexibility during the interview (Barbara B.Flynn et al. 1990).

The product catalogue of the company was also studied to get the additional data for the project thesis. All data collection methods and channels were fully authorized by the company management and it was carried out without transgressing any ethical rules of the company.

3.5 CHOICE OF PRODUCT FOR INVENTORY MANGEMENT STUDY

The company produces more than fifty types of valve assemblies of different valve types, gate valve, ball valve, globe valve, check valve etc. Among this wide product range, purposive sampling was used to identify and select 6 product items for the study. The sales report and purchase orders history was retrieved from the company's information system. The fast moving items from the products are selected which are ordered in large numbers so that representative items are used for research study.

3.6 DATA COLLECTION

Extensive data collection campaign was carried out with a broad perspective of following details related to inventory management.

- Historical annual demand of the major product range, 6 product items were selected through purposive sampling.
- Finished goods inventory maintained for each of the product
- Work in process inventory.
- Bill of Material for each of valve assembly.
- Company inventory policy.

- List of important raw material and component assembly with the list of suppliers (multiple suppliers if any).
- Ordering cost and inventory carrying cost of each of the product item.
- Lot size, reorder point and safety stock of each of the product item.
- Quantity discounts offered by the supplier and price breaks.
- Delivery lead time for each of the component and history.
- Quality of received components and rejection rate.
- Raw material and finished goods storage capacity restrictions.
- Bidding policy of the company.
- Procurement philosophy of the company.

Relevant data for the AI modeling was based on the importance of selected factors.

3.6.1 Sales data

Past historical bimonthly sales data for these product categories were compiled. This data formed the time series for forecasting the demand for these types of valves. Sample Bi-monthly Sales data for one of the product item under study, 10''X 150 GTV 101 is shown in the Table 3.1. It is considered not necessary to study the different influencing factors for the demand prediction, as the time series itself is the reflection of aggregate response of the variable under consideration to the different factors.

The company has got a historic data regarding the on time delivery performance and quality rejection rate of the established vendors which helps to fix the reserve stock. The reserve stock will help to meet the sudden rush demand and act as buffer against the delayed deliveries from the vendors.

3.6.2 Ordering cost data

Ordering cost is the administration cost of ordering the material. This cost is incurred every time the material is ordered. It consists of two parts.

Table 3.1 Sample Bi-monthly Sales Data for 10''X 150 GTV 101

Year	Month	Domestic sales Qty (Nos)
2001	Jan-Feb	48
	Mar-April	64
	May-June	52
	July-Aug	35
	Sept-Oct	55
	Nov-Dec	70
2002	Jan-Feb	65
	Mar-April	63
	May-June	76
	July-Aug	66
	Sept-Oct	40
	Nov-Dec	70
2003	Jan-Feb	30
	Mar-April	42
	May-June	40
	July-Aug	44
	Sept-Oct	55
	Nov-Dec	40
2004	Jan-Feb	58
	Mar-April	60
	May-June	62
	July-Aug	70
	Sept-Oct	55
	Nov-Dec	42
2005	Jan-Feb	55
	Mar-April	54
	May-June	76
	July-Aug	36
	Sept-Oct	39
	Nov-Dec	90
2006	Jan-Feb	68
	Mar-April	66
	May-June	45
	July-Aug	72
	Sept-Oct	73
	Nov-Dec	56

The cost of the ordering process itself: This is the fixed cost and does not depend on the number of units ordered. It includes the clerical cost of purchase function like vendor identification, invoice processing, communication and accounting related expenses. This part of ordering cost per unit will reduce as we order more units per order

The inbound logistics costs: This refers to cost involved in transportation, loading& unloading and inspection. Those costs may be variable. Thus the ordering cost can be different for different products depending on the distance of supplier from the point of delivery, type of transportation and containerization used, type of requirement of inspection methods (Hans-Joachim Girlich, 2003).

The cost data maintained by the company records is referenced and is used in the lot size calculation.

3.6.3 Inventory carrying cost data

There are costs incurred for carrying all inventories. Accounting and economic cost form the two components of carrying cost. Accounting costs are obvious costs and they are actual cash payments. Economic costs are abstract costs and hidden like opportunity costs (Goldsby et al. 2005). The finer constituents of inventory carrying costs are illustrated in Figure 3.3.

3.6.3.1 Capital costs

WACC (weighted average cost of capital) is the usual approach to calculate the capital cost. This turns out to be the rate which the company pays on average to all its security holders to finance its assets. Capital cost comprises of the interest on working capital and the opportunity cost which means the difference between the returns when the same capital is invested in inventory and in other comparable assets like treasury or mutual fund.

3.6.3.2 Storage space costs:

These are the expenses incurred on building and maintenance of the facility for storing the inventory. The cost of maintaining basic facilities like air conditioning, heating, lighting will form a part of the storage space cost. Notional expenses of depreciation will also contribute to storage space cost. Expenses on property tax and lease, cost of purchase also need to be taken into consideration under this head. Company owned or rented property for storage will greatly influence this cost. The company faces the problem of saturation or near saturation of storage space which is increasing its storage cost non linearly. This is caused due to the obstruction for the free movement in an almost filled warehouse and increased difficulty in finding alternate storage space or extra emergency storage space .

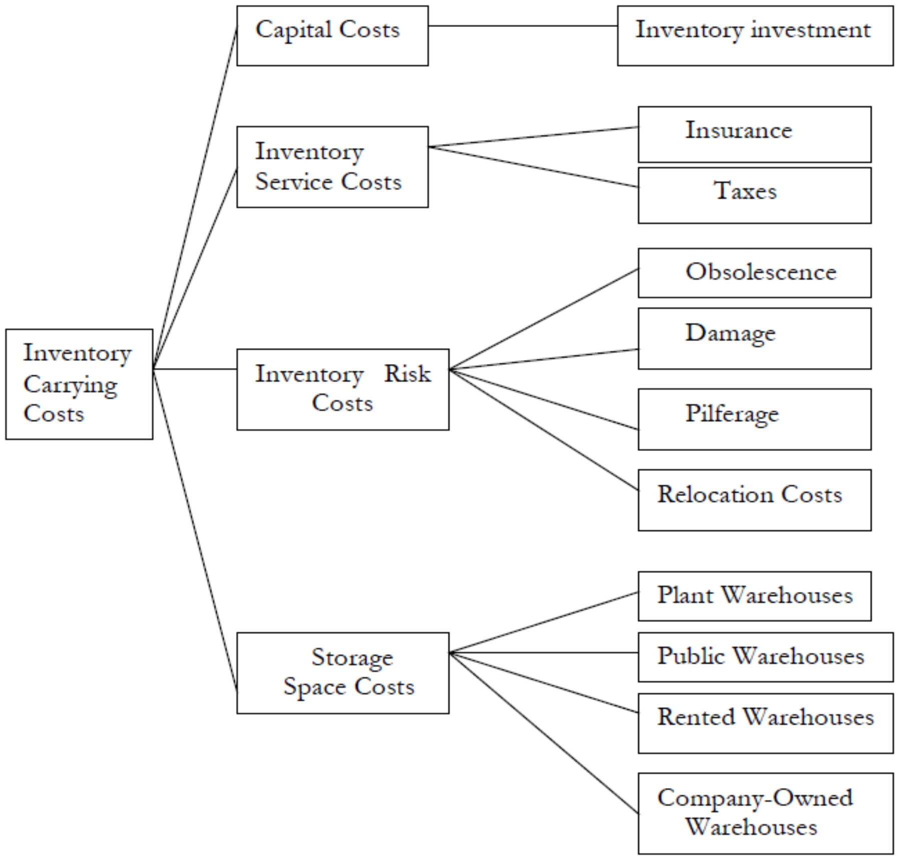


Fig 3.3 Components of Inventory Carrying Cost (Source: Leenders et al. 1985)

3.6.3.3 Inventory services costs

These are costs involved in system hardware and software installed for the inventory management, RFID if implemented, manual material handling for servicing the inventory. Cost involved in cycle counting for physical stock verification is also a part of his cost.

3.6.3.4 Inventory risk costs

This cost includes the degradation in value of the inventory over the period of storage. This may be the result of the product shrinkage, theft, fraud at different points of purchase like the vendor himself or stores etc. Administrative errors like misplaced goods, shipping errors are also covered under this head. Obsolescence, outdated products, damages in transit will also add to inventory risk cost.

Inventory Carrying Costs in Summary:

Cost of Money 5% - 11%

Taxes 2% - 5%

Insurance 1% - 3%

Warehouse Expenses 2% - 6%

Physical Handling 3% - 6%

Clerical & Inventory Control 2% - 5%

Obsolescence 5% - 15%

Deterioration & Pilferage 2% - 5%

Total 20% - 50%

Inventory carrying costs for the product items chosen are taken from the company records and used for the calculation in the multi objective optimization study.

The above compiled data is used for solving the lot sizing problem which covers the second objective of research. Lot sizing decision comprises of determining how much of a product is to be produced at what time so that the total cost is minimized, at the same time responding well for the demand requirements under the available capacity. Estimating the right lot sizes affects inventory cost and also the service level and customer satisfaction.

3.7 NEURAL NETWORK MODEL

Neural networks are flexible data driven models that have appealing properties for prediction. Fig. 3.4 shows the work flow in neural network design.

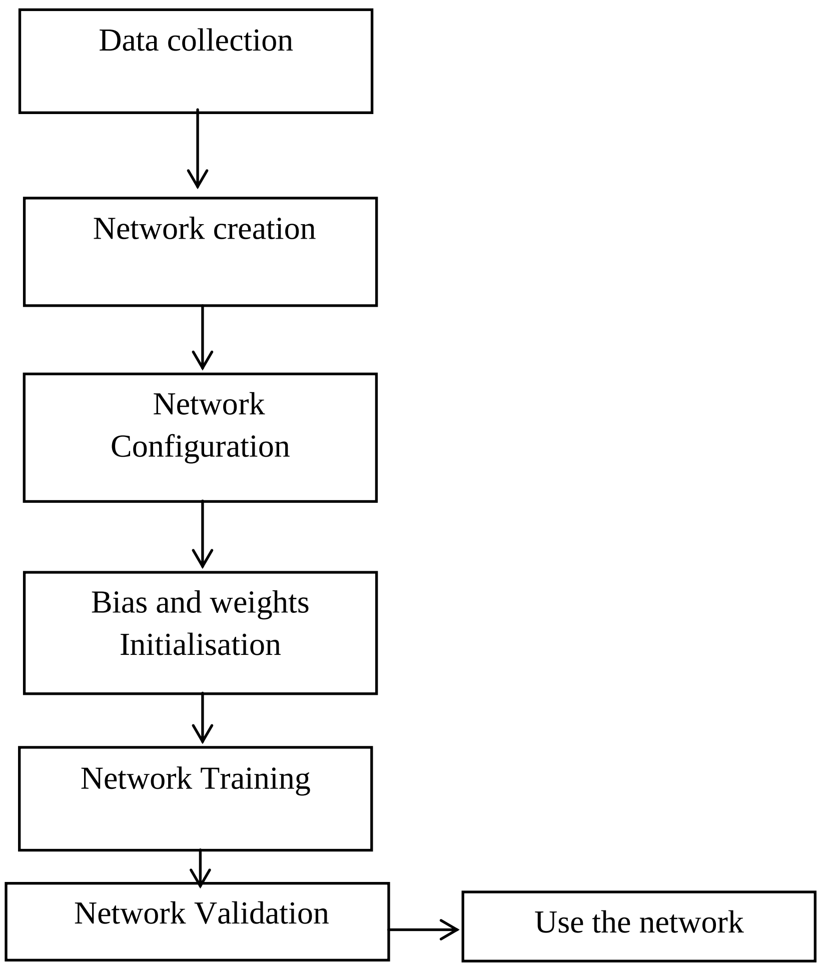


Fig 3.4 Work flow in neural network design (Garetti, M. et al. 1990)

Neural networks are universal approximators and are well known to capture the non-linear relationship thus improving the performance of the forecast (Hill,T. et al.2006). The aim of the study is to build a neural network model for the demand forecast of the industrial valves. There are two models developed for prediction, namely, Multilayer Perceptron (MLP) and Radial basis Networks (RBN). The learning algorithm used in the both study are Error back propagated Gradient Decent method. The following section will focus on the architecture and training methods used in the present work.

3.7.1 Architecture of Multi-Layer Perceptron model

The complex engineering and business application related problems have been successfully resolved by using ANN with the Multilayer perceptron (MLP) architecture (Singh, P. et al. 2007). These problems are successively solved by machine learning with an effective error back propagation algorithm. The errors occurred during training are fixed using different learning rules. Least Mean Square algorithm and Gradient Decent are some of them. The architecture contains three types of layers. They are named input layer, hidden layer, and output layer. Fig 3.5 represents the multilayered ANN structure. The definition of the architecture determines the different parameters like the number of layers in the network, the number of neurons in each layer, the transfer function of each layer and the layers interconnectivity. The best architecture is chosen based on the type of the problem that network represents.

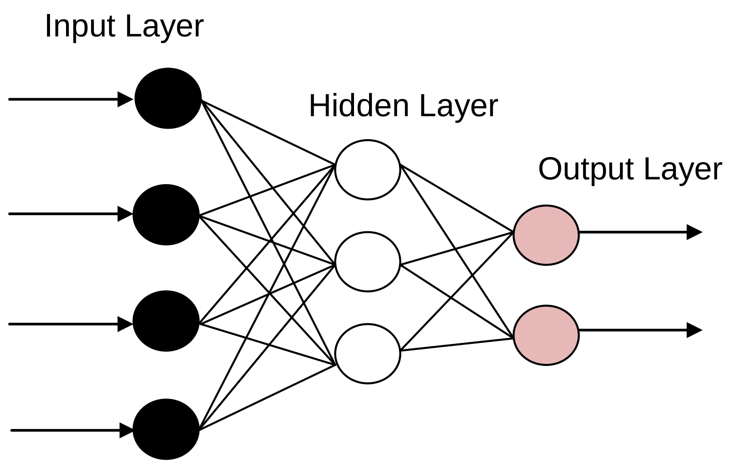


Fig 3.5 Multilayered ANN structure (Courtesy: Simon Haykin,1999)

The flow of signal is carried from input layer to output layer in a forward direction, thus it is named as Feed Forward Neural Network. The supervised learning involves the process of adjusting the network, so that a particular input or a group of inputs result in a specific target output. This process is called training the network. Network training compares the output and the target until the output matches the target. A number of input and corresponding target outputs are fed to the system as training data. During the process of training, the relationship between the input and output is studied. The synaptic weight adjustment process continues as the learning process progresses. Fig. 3.6 explains the learning process (Herbrich, R. et al. 2000).

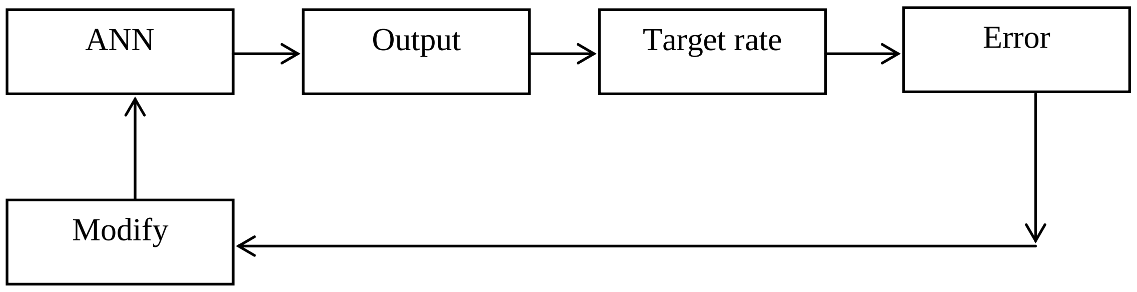


Fig 3.6 A Training process in an ANN model (Courtesy: Herbrich, R et al. 2000)

The configured neural network has to work in such a way that the set of inputs should provide anticipated result. The weights are basically set in two ways. One method is to set the weights based on the previous knowledge. The other way is by providing the data as a train to learn the pattern of output by means of learning rules. Networks must be trained in such a way that the errors between the desired and target are minimal. There are two types of error specification, by specifying the number of epochs and by specifying the error value. In epoch specification, the training data will run up to the specified number of epoch and once it reaches specified value, the testing of the data is carried out. In the case of error criterion, the training iteration will run till the error is minimised to specified value. In the present study, MLP architecture is used for training and predicting sales demand of industrial valves. Inputs to the neural network are actual demand for the immediate last period, moving average of last two periods, three periods and six periods. Output or predicted parameter is the demand forecast for next period.

3.7.2 Back Propagation Training Algorithm

Mapping of input and output relationships through multi layer networks for the proper training of ANN is achieved by Back propagation algorithm. The essential working process of back propagation consists of two passes through different layers of the network. In the forward computation, the input data is acquired and propagated through network from one layer to another layer. The network responds by generating the output set. Forward pass fixes all the connection weights. Backward pass of the algorithm adjusts the weights base on the error between actual and target output. The process of forward and backward pass persists till the learning is complete with adequate accuracy or till the overall error is reduced to acceptable level. The error gets accumulated over the entire training cycle. This computation cycle is called training epoch. After the training phase, it will be the testing phase. During this time the trained network whose synaptic weights were fixed during training phase, operates with forward pass or normal feed forward algorithm to give the output (James A. Anderson1995).

Different back propagation algorithms are used to train the neural network depending on the architecture. The most used is Levenberg-Marquardt (trainlm**),** which is ideal for training the small and medium sized problems (Wong, B. K. et al. 2007).

3.8 RADIAL BASIS FUNCTION NEURAL NETWORKS

This section presents an overview of radial basis function neural networks and their application for prediction of product demand (Moody et al. 1989).

Radial basis function (RBF) networks are a class of feed-forward networks. Supervised training algorithm is their learning platform. The network configuration in case of RBF networks normally consists of a single hidden layer of units. The triggering of these units is from a class of functions called basis functions. RBF networks inherit all the basic characteristics and advantages of back propagation. They even score several plus points over Multi layer perceptron. RBF usually train and converge more quickly than back propagation networks. Non-stationary inputs will not adversely affect the performance of this type of network because of the behavior of the radial basis function hidden units.

RBF models are considered superior due to their simplicity and ease of implementation (Srinivasa Pai.P, et al. 2004). These networks have got superb learning and function approximation capabilities. A radial basis function network in its simple form consists of three different layers. Source nodes or sensory units form the input layer. The second layer is the hidden layer. Third layer is output layer which gives the response to the activation patterns applied to input layer. From input space to the hidden-unit space, the transfer function is non linear, whereas the transformation from the hidden-unit space to the output space is linear. Fig 3.7 shows the typical RBF architecture. In this figure, p_1 to p_n represent the input and y represents the output.

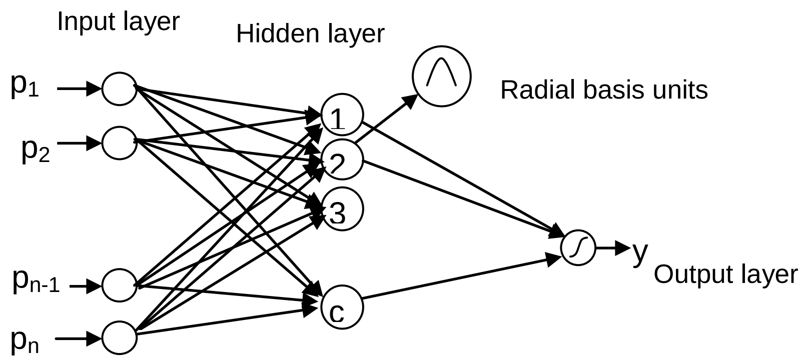


Fig 3.7 General architecture of RBF network (Simon Haykin, 1999)

Main advantage of RBF network comes from their capability in universal approximation. Other important benefit of RBF network application is that learning requires very less computation time and it produces smaller network than other algorithms (Simon Haykin, 1999). Gaussian function is selected as basis function in the standard approach to RBF network implementation. Input data characteristics govern the selection of number of hidden units. Normal least square method is used to estimate the weights between hidden and output units as they are linear.

Moody and Darken (1989) conducted in depth research into RBF networks and have proven the usefulness of neural network architecture. Significant variance between RBF and MLP is caused due to the behavior of the single hidden layer. Activation function used in MLP back propagation is sigmoidal or S-shaped. Gaussian or some other basis kernel function is the activation function used by hidden units in RBF. Each hidden unit behaves as a locally adjusted processor.

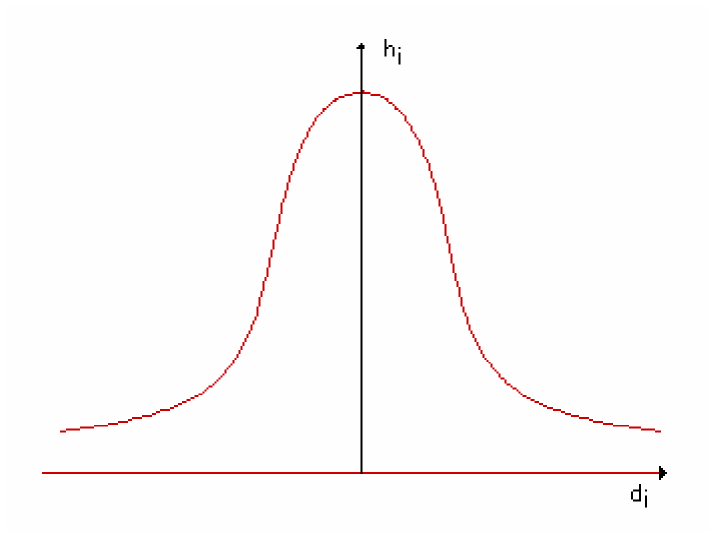
Broomhead and Lowe (1988) pioneered the application of radial-basis functions to the design of neural networks. Papers presented by Moody and Darken (1989), Poggio and Girosi (1990), Simon Haykin (1999) and Renals (1989) were the other major works in the field of design and application for RBF neural network.

3.8.1 The Structure of the RBF Networks

Radial Basis Function was first applied in the solution of interpolation problems for real multivariable fnctions. Broomhead and Lowe (1988), and Moody and Darken (1989) are credited with the first deployment of the radial basis functions in the design of neural networks. The basic form of RBF network consists of three different layers, input, output and hidden layers.

The input layer contains the source nodes. The number of source nodes is determined by the dimension 'm' of the input vector '**s'.** Hidden layer of nonlinear units constitutes the second layer. These nonlinear units are linked straight to all the nodes in the input layer.

Each hidden unit consists of a basis function. The basis function is characterized by two parameters, center and width. Graphical presentation in fig 3.8 shows the basis function curve. The width of basis units is determined by the variance, $\sigma_{i.}$ of the basis function. The basis function has highest value at zero distance and it reduces as the distance from the center rises.



d_i is Radial distance of input vector U from the center of basis function.
h_i is the output of each hidden unit i

Fig. 3.8 Graphical representation of Radial Basis Function (Courtesy: Moody et al. 1989).

The hidden units are connected to the input units by non linear transfer functions. The output layer is connected to hidden layer by linear transfer functions.

The j th output is computed as

$$x_j = f_j(u) = w_{oj} + \sum_{i=1}^{c} w_{ij}\, h_i \qquad Eq. 3.1 \qquad [\text{Moody and Darken, 1989}]$$

Where j = 1, 2,…m , i= 1, 2, …c

The centers of RBF units have to be fixed using different approaches and their learning characteristics will be analyzed. Performance of RBF neural networks to be compared with MLP for effectiveness in prediction.

The RBF network is a single hidden-layer feed forward neural network. Each node of the hidden is characterized by two parameters, a center x_j and a width σ_j. The parameter center is responsible for the radially symmetrical response of network input vectors. Interpolating function smoothness property is influenced by the width of the network.

3.8.2 RBFNN training strategies

RBF networks are trained on different learning strategies based on the specification of centers of radial basis functions. Two such different learning strategies are discussed below.

3.8.2.1 Fixed Centers Selected at Random

Activation functions of hidden units are defined by fixed radial basis functions in the simplest approach for RBF neural network. Centers are selected on random basis from the training data set. The RBFs use Gaussian activation function which is defined as

$$\Phi_j\,(x) = e^{\frac{-||x_j - p_i\,||^2}{2\sigma_j{}^2}} \qquad \text{Eq. 3.2} \qquad \text{(Matlab, 2008)}$$

where x_j is the center and σ_j is the width (standard deviation), $j = 1, 2 \ldots c$ where c is the number of centers, p_i is the input value and $\Phi_j(x)$ is the activation function. Linear weights in the output layer of the network will be fixed during learning process. This is the only parameter which is evaluated during learning. LMS approach or gradient descent approach is used for network learning (Beale et al. 2000).

3.8.2.2 Self-Organized selection of centers.

This is another approach for network learning strategy where the radial basis functions locate their centers in a self organized design. A supervised learning rule is used to compute output layer weights. The network fixes its parameters by hybrid learning process (Prasanna Kumar et al. 2013). The network resources will be apportioned in a highly planned process by locating the centers of radial basis functions in those sections of input horizon where significant data are present. Clustering algorithms like fuzzy c-means (FCM) and its modified versions are used to achieve the self organized selection of centers.

After the center selection, next phase of training is the evaluation of width of radial basis units $\sigma_{j.}$ Different heuristics can be applied for the computation of width parameter. One of the simplest method is to choose all the σ_j to be equal. This method ensures a smooth distribution of training data caused by the overlap of basis functions.

Another convenient and more effective method is to find the widths is the P-nearest neighbor heuristic (Moody & Darken, 1989). In this method, the width parameter σ_j is given by the root mean square distance of the given cluster center to the P- nearest neighboring centers. That is

$$\sigma_j = \sqrt{\left(\frac{1}{p}\sum_{p=1}^{p} \| x_j - x_{jp} \|^2\right)} \qquad \text{Eq. 3.3} \qquad \text{(Hadjahmadi et al. 2008).}$$

Where vector x_j ($j = 1\ldots.c$) are the centers and $x_{j1}, x_{j2}, \ldots. x_{jp}$ ($1 <= j_1, j_2, \ldots..j_p < = c$) are the P nearest neighboring centers.

3.9 GENETIC ALGORITHM

Genetic algorithm is a population based meta-heuristic which can be utilized to solve complex optimization problems. John Holland invented genetic algorithm which mimics the principle of natural genetics in order to solve optimization problems. (John Holland, 1962). Genetic algorithm uses knowledge from previous generations to direct the future search (John Holland, 1975). A number of chromosomes at once rather than a single chromosome can be used to carry out the search process. The optimum or near-optimum solution of optimization problems can be efficiently obtained using genetic algorithm (John Holland, 1962). The flow chart of genetic algorithm to solve inventory control problem is shown in Fig. 3.9

The GA algorithm has been implemented on JAVA platform and the program can be run with Net Beans IDE. The following components of genetic algorithm have been implemented to solve the problem.

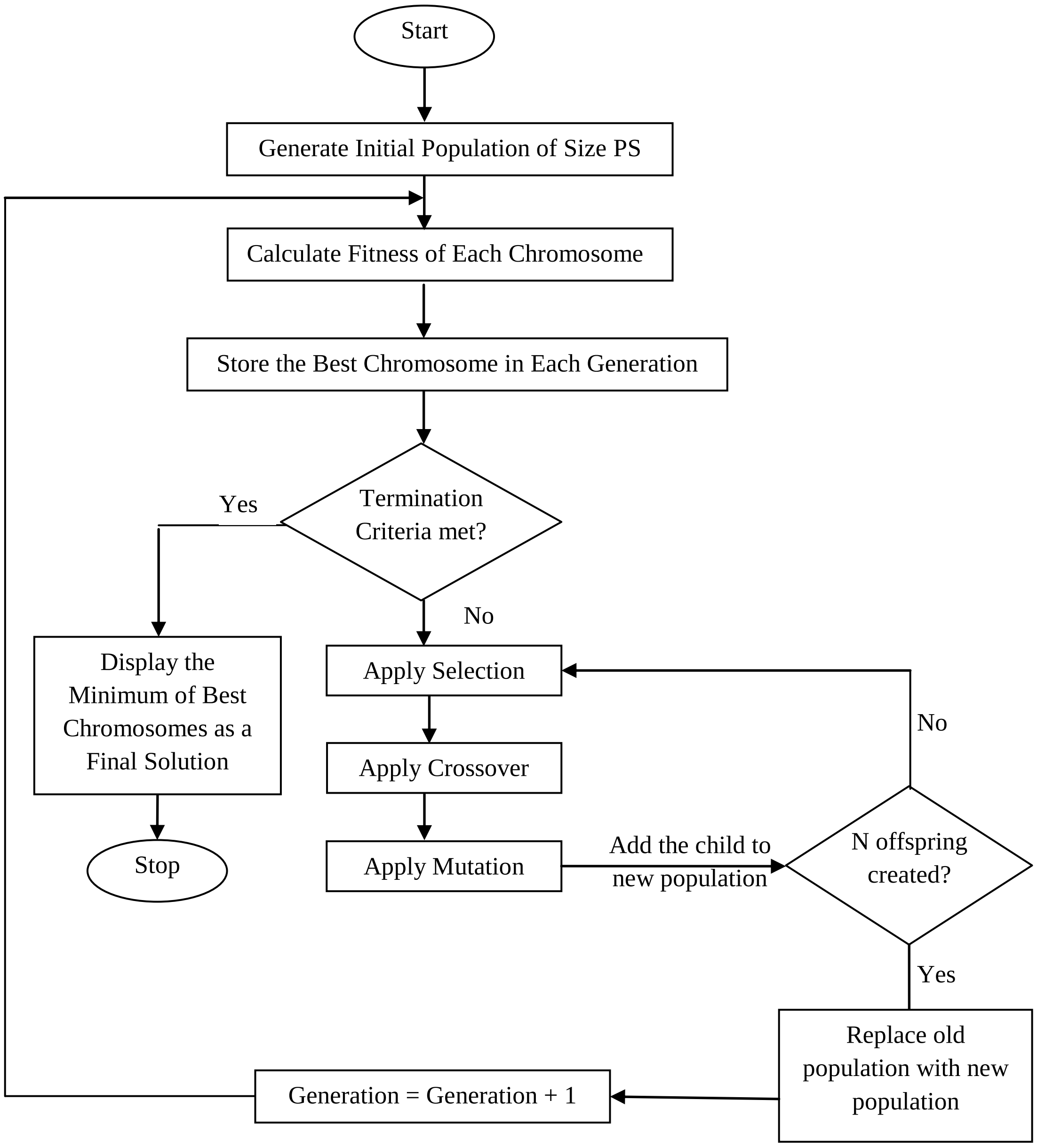

Start
Generate Initial Population of Size PS
Calculate Fitness of Each Chromosome
Store the Best Chromosome in Each Generation
Termination Criteria met?
Yes
No
Display the Minimum of Best Chromosomes as a Final Solution
Stop
Apply Selection
Apply Crossover
Apply Mutation
Add the child to new population
N offspring created?
No
Yes
Replace old population with new population
Generation = Generation + 1

Figure 3.9 Flow chart of proposed genetic algorithm

3.9.1 Chromosome Representation

The chromosomes represent the strings of the order quantities of the items in each period in this research work. For example, the chromosome representation matrix (CR) with four items and three periods is shown in Fig. 3.10. In Fig 3.10, the number of items and periods are represented by rows and columns, respectively.

$$CR = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{21} & Q_{22} & Q_{23} \\ Q_{31} & Q_{32} & Q_{33} \\ Q_{41} & Q_{42} & Q_{43} \end{bmatrix}$$

Figure 3.10 Chromosome representation (Seyed Mohsen Mousavi et al. 2013).

3.9.2 Initial population generation

Initial population of size PS is generated by using problem specific knowledge so that it does not violate the constraints 3, 4, 5 and 6. Population size (PS) can be increased or decreased based on the input size of the problem. Population size is determined by Taghuchi method in this research work (Seyed Hamid Reza Pasandideh et al. 2011).

3.9.3 Calculate fitness of each chromosome

The process of calculating the fitness of each chromosome consists of the following two steps.

1. Calculate the objective function of each chromosome.

2. Calculate the fitness of each chromosome based on its objective function value using the following equation.

$f(x) = 1/(1+O(x))$ Eq. 3.4 **(**Seyed Mohsen Mousavi et al. 2013)

where, f (x) is the fitness function of each chromosome and O(x) is the objective function of each chromosome.

3.9.4 Selection operator

The roulette wheel selection is used to select the chromosomes. In roulette wheel selection, chromosomes are selected into the mating pool according to their raw fitness. The chromosomes having high fitness have more chances to be selected. The probability of i^{th}-selected chromosome is

$$P_i = \frac{f(i)}{\sum_{j=1}^{n} f(j)} \qquad \text{Eq. 3.5}$$

where, f (i) and f (j) are the fitness of the chromosomes i and j respectively (Patel, J.N. 2011).

3.9.5 Crossover operator

Single point crossover is used in this research work. The following Figure 3.11 demonstrates the application of single point crossover on the two selected parents.

$$\text{Parent } 1 = \left[\begin{array}{cc|c} Q_{11} & Q_{12} & Q_{13} \\ Q_{21} & Q_{22} & Q_{23} \\ Q_{31} & Q_{32} & Q_{33} \\ Q_{41} & Q_{42} & Q_{43} \end{array}\right] \qquad \text{Offspring } 1 = \left[\begin{array}{cc|c} Q_{11} & Q_{12} & O_{13} \\ Q_{21} & Q_{22} & O_{23} \\ Q_{31} & Q_{32} & O_{33} \\ Q_{41} & Q_{42} & O_{43} \end{array}\right]$$

»

$$\text{Parent } 2 = \left[\begin{array}{cc|c} O_{11} & O_{12} & O_{13} \\ O_{21} & O_{22} & O_{23} \\ O_{31} & O_{32} & O_{33} \\ O_{41} & O_{42} & O_{43} \end{array}\right] \qquad \text{Offspring } 2 = \left[\begin{array}{cc|c} O_{11} & O_{12} & Q_{13} \\ O_{21} & O_{22} & Q_{23} \\ O_{31} & O_{32} & Q_{33} \\ O_{41} & O_{42} & Q_{43} \end{array}\right]$$

Figure 3.11 Single point crossover (Seyed Hamid Reza Pasandideh et al. 2011)

The offspring may not satisfy the constraint 3 of inventory status as specified in section 5.2 of mathematical model formulation. If, any, offspring does not satisfy the constraint 3, then the correction step is applied after the crossover operation in a way that it will satisfy the constraint 3. Furthermore, crossover rate is taken as 0.9 in order to protect some of good chromosomes that are already present in the mating pool.

3.9.6 Mutation operator

A novel mutation operation, namely “subtraction-addition” mutation is used in this research work as follows (Gupta,R.K. et al. 2009):

Step 1: Any one period is selected randomly for each component.

Step 2: Mutation value is randomly generated.

Step 3: If selected period is t then mutation value is subtracted from period t and added to period T. If selected period t is the last period then T is the first period. If selected period t is not the last period then T = t + 1 period.

This novel mutation operation never violates the constraint 3. Again the mutation value is generated in a way that this novel mutation operation never violates the constraint 4 too. Moreover, mutation rate is taken as 0.4 in order to escape from local optima.

3.9.7 Replacement strategy

The chromosomes in old population are replaced by newly generated chromosomes which meet both the budget constraint 5 and warehouse area constraint 6 in each generation. The newly generated chromosomes which do not meet either the constraint 5 or constraint 6 will be discarded.

3.9.8 Stopping criteria

Genetic algorithm stops when it has completed maximum number of generations. Maximum number of generations can be increased or decreased based on the input size of the problem. Optimum number of generations to be used is an important GA parameter which is determined using Taghuchi design in this research work.

3.10 ANT COLONY OPTIMISATION.

NP-hard combinatorial multi objective optimization problems can be solved by different Artificial Intelligence Algorithm. Ant Colony Optimization (ACO) is one such approach.

It is a meta-heuristic population based method (Dorigo et al. 2010). This approach mimics the natural process of the indirect communication which takes place between real ants by means of trails of a chemical substance called pheromone. Simple agents called artificial ants are used in problem solution. They use the information regarding the particular problem which is analogous to artificial pheromone information, to share their experience to other ants in the problem solution population. This is the basis on which all ant algorithms approach for multi objective optimization works. Fig. 3.12 shows the flow chart of proposed ant colony optimization algorithm

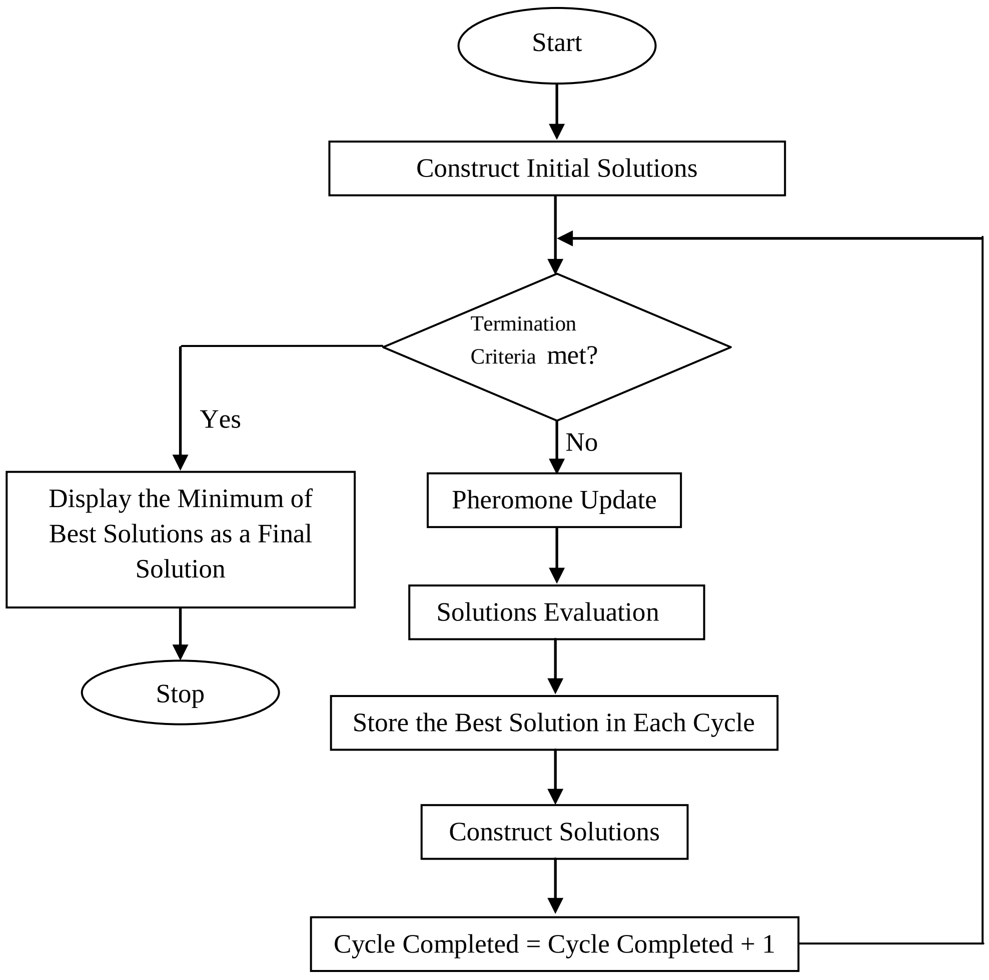

Start
Construct Initial Solutions
Termination Criteria met?
Yes
No
Display the Minimum of Best Solutions as a Final Solution
Stop
Pheromone Update
Solutions Evaluation
Store the Best Solution in Each Cycle
Construct Solutions
Cycle Completed = Cycle Completed + 1

Fig. 3.12 Flow chart of proposed ant colony optimization algorithm (Coutersy: J.J. Patel et al. 2011).

Ant cycle model is used in this research work. In this model, the trails are globally updated during each cycle by all ants. The amount of pheromone deposited by each ant is a function of the solution quality. As per the flowchart of proposed ant colony optimization algorithm, first the ant based initial solution construction method is executed.

Next values of pheromone are set based on ant based initial solution construction method. Thus the values of pheromone are set accurately after the execution of ant based initial solution construction method. Next ant based solution construction method will execute in each cycle which will utilize the values of pheromone in order to construct better solutions (Colorni et al. 1991).

The novel concept is implemented for ant based solution construction in each cycle as follows. First each ant will construct the solution according to the probabilistic rule. After constructing the solution according to the probabilistic rule, ant 1 will discard its order quantity values of all periods for item 1 and item 3 and the values of order quantity of all periods for item 1 and item 3 of ant 2 will be copied into the solution generated by ant 1. Similarly, ant 2 will discard its order quantity values of all periods for item 1 and item 3 and the values of order quantity of all periods for item 1 and item 3 of ant 3 will be copied into the solution generated by ant 2. This discard-copy step is repeated for all ants. Last ant will discard its order quantity values of all periods for item 1 and item 3 and the values of order quantity of all periods for item 1 and item 3 of ant 1 will be copied into the solution generated by last ant. This novel concept represents the direct communication between ants in each cycle in order to further improve the solution constructed by individual ant separately (J.J.Patel et al. 2011).

In this research work, ant based solution is constructed by using problem specific knowledge so that it does not violate the constraint 3 and constraint 4, specified under mathematical modeling under section 5.2. Once the value of order quantity is generated for all components for all periods, it will be verified that whether the generated values of order quantity will meet the budget constraint and warehouse area constraint or not. If either of the constraints is not met then the generated values of order quantity (solution) will be discarded. If both the budget constraint and the warehouse area constraint are met then the solution is valid and it is stored for further processing.

The ACO (Ant Colony Optimisation) algorithm has been implemented on JAVA platform and the program can be run with Net Beans IDE. (Integrated Development Envionment)

3.11 TAGUCHI DESIGN

The Taguchi method is one of the commonly employed approach used to optimize a response using some designed experiments that are performed based on different combinations of some controllable factors (Javad Sadeghi et al. 2014). Taguchi studied and devised fractional factorial designs of experimentation to reduce the large number of experiments in full factorial design (Najafi, A. et al. 2009).

In this work, the Taguchi method is used to tune the parameters of meta-heuristic algorithm of multi objective optimization, genetic algorithm. The parameters of a meta-heuristic that are needed to be tuned act like controllable factors in the design of experiments (DOE). The aim is to find an optimal combination of the parameters such that the response (the fitness function) is optimized. Consequently, a set of experiments is performed in this section and the results are statistically analyzed. Design of experiment consists of two or more parameters, with different sets of levels. Each level is varied in statistical manner. The outcome of the particular test combinations is observed, and the complete set of results is analysed to determine the influencing factors.

The Taguchi method is a special case of the fractional factorial design in which some special orthogonal arrays are used. In order to study the results acquired by the Taguchi method, two approaches are suggested in the literature. First, for experiments with single run, application of the analysis of variance is advised. For experiments that are carried out on multiple runs, a response variable known as Signal to Noise ratio (S/N) is suggested. In the above relation, S denotes factors which are controllable; N represents noise elements affecting response (R. Roy 1990). Since the meta-heuristics need to be run number of times to acquire a better solution, in this research we use S/N to analyze the results using Taguchi et al. (2005).

Many statistical tools are available to optimize the control parameters. But orthogonal arrays under Taguchi method would help to study a large number of decision variables with a limited number of experiments. Decision variables are divided into controllable and noise factors. Noise factors can not be controlled directly. It is also impractical and most of the time impossible to eliminate the noise factors (Phadke MS, 1989). Taguchi experimental design will help to reduce the effect of noise factors.

3.11.1 Signal to Noise Ratio (S/N ratio)

Taguchi adopted the concept of signal to noise ratio to reduce the effect of noise factors in the experiment. The desired value or mean response value is represented by signal. The undesirable value or standard deviation is denoted by noise. The variation present in the response variable or the component of noise factor is represented by S/N ratio (Ross RJ, 1989). Objective functions are classified into 3 types for design of experiment applications by Taguchi. They are “smaller the better”, the “larger the better”, and “the nominal is best.” Since almost all objective functions in inventory control systems are grouped in the “smaller the better” type, its corresponding S/N ratio is:

$$\frac{S}{N} = -10\ log_{10}(Objective\ function)^2$$ (Syed, M.M et al. 2013).

3.11.2. Taguchi method implementation

Taguchi method implementation consists of five basic steps (R.Roy, 1990).

- Parameters with significant effects on the response are determined.

- The parameter value is determined by the trial and error procedure so that good fitness value is obtained while implementing the experiments.
- Taking into consideration the available degree of freedom (DOF), a suitable orthogonal array is selected which would specify the number of experiments to minimize the experimentation time, and at the same time would be giving full weightage for all the combination of influencing factors.
- Experiments are conducted based on obtained design.
- The results are recorded. The S/N approach will give the basis for evaluation and analysis of the results.

3.12 ANALYSIS OF VARIANCE (ANOVA)

Analysis of variance (ANOVA) is a statistical technique which can be applied to analyse the differences among different population means. ANOVA is used to study the variation among and between groups. ANOVA computation partitions the variance of a particular variable into different components which can be assigned to different causes of variation. ANOVA provides a basis of a statistical tool which confirms if the means of several population are equal (Seyed Hamid et al. 2010).

ANOVA has got tremendous application in the analysis of experimental data as a tool for testing statistical hypothesis.

When there are two or more than 2 groups, ANOVA is the tool which determines, if there are any, statistically significant differences between the means of groups. In this research work, one way ANOVA is used to compare the results obtained by the different meta heuristic algorithm for the solution of the mathematical model for the multi item multi period periodic lot sizing in inventory control analysis. Software Mini tab 15 is used for the computational purpose.

As a statistical computation instrument, one way ANOVA compares the means between groups to conclude if any of these means are significantly different from each other. Effectively, it examines the null hypothesis.

where μ = group mean and k = number of groups. On the other hand, if the one-way ANOVA gives a result of significant different means, the alternative hypothesis (H_A), is accepted which means that there are at least 2 group means that are significantly different from each other (Amy H.I. Lee et al. 2013).

3.12.1 ANOVA table analysis

ANOVA analysis yields the ANOVA table as its main output irrespective of the software used for computation Anova table is characterized by number of columns with the following labels in order: (Table 3.2)

- source of variation
- Sum of squares
- Degree of freedom
- Mean square
- F- value
- P value

The rows are labeled 'Between Group variation', 'Within Group variation'. There is an additional row containing total variation (Javad Sadeghi et al.2014).

$$\text{Mean square} = \frac{\textit{Sum of Squares}}{\textit{Degree of freedom}}$$

Single F statistic is placed in the' between groups row' and is calculated using formula:

$$F == \frac{MEAN\ SQUARE\ between}{MEAN\ SQUARE\ within}$$

P value is obtained by comparing the F value to its null sampling distribution.

The F-statistic tends to be greater if the alternative hypothesis is true or if the null hypothesis is false. We reject the null hypothesis if $p \leq \alpha$. Where α is the confidence level.

Table 3.2 Structure of results of ANOVA table.

Source of Variation	*Sum of Squares*	Degree of Freedom	*Mean Square*	*F value*	*p*
Between groups	SS_b	k-1	MS_b	MS_b/MS_w	*p* value
Within groups	SS_w	N-k	MS_w		
Total	$SS_b + SS_w$	N-1			

(Source: Javad Sadeghi et al.2014)

3.13 SUMMARY

To summarize, the research methodology adopted is explained in detail in this section. Inventory management of company under study has been analysed. Data collection method has been briefly explained. AI approaches of Artificial neural network with its different architecture of MLP and RBFNN are described with respect to their application for demand forecast. ACO and GA Modeling of multi objective optimization inventory problem for period lot sizing have been elucidated. Application of Taguchi method of Design of Experiments for tuning the parameters of GA has been explained. Concept of one way ANOVA analysis has been detailed which will be used to compare the performance of GA and ACO algorithm for inventory control problem. Results and discussion have been presented in the subsequent chapters.

Chapter 4

4. RESULTS AND DISCUSSION – PART I

ANN APPROACH FOR DEMAND FORECAST

4.1 INTRODUCTION

With the objective of attaining better accuracy for the demand forecast, two different architectures of artificial neural network modeling have been discussed namely Multi Layer Perceptron and Radial Basis Function Neural Network. The periodic demand data generated out of resulting demand forecast has been used for constructing GA and ACO models for optimizing dynamic lot sizing for multi period multi item periodic reorder. The results of the studies are presented and discussed in-depth in this chapter.

Detailed discussion has been carried out under the Research methodology chapter regarding the versatility, importance and wide ranging application of ANN in prediction related problem solution.

Multilayer perceptron (MLP) and Radial basis functions (RBF) are the two types of feed forward neural networks which have been utilized for modeling of demand forecast in the current research work. Section under Research Methodology deliberated extensively on these two architectures. In the following sections, a systematic and detailed review has been presented regarding the application of the important theoretical concepts regarding the MLP, RBFNN and back propagation algorithm to build the ANN model for the demand forecast.

4.2 MLP architecture for ANN model for demand forecast

MLP has already been introduced in the section 3.2 as one the most used ANN architecture (Simon Haykin 1999). We have seen that Back propagation algorithm is one of the important training methods for MLP [Herbrich, R. et al. 2000, Singh, P. et al. 2007]. It was also described how the Back propagation algorithm performance can be improved by adjusting different factors like rate of learning, initial weight, transfer function of nodes, presentation training data, momentum coefficient, number of neurons in each layer, training algorithm etc. (Wong, B. K. et al. 2007).

4.2.1 Identification of best training Method:

In this research study, a preliminary ANN model was developed by using MATLAB 8 (2008) to evaluate the different training methods. On confirmation of Trainlm as the best training method, a robust demand forecast ANN model was developed on C++ platform using this training method.

Preliminary ANN model was tested using the standard training functions available in the neural network tool box. The MLP network is trained using the following training algorithms [Tugba Efendigil et al.2009, Koskivaara, E. et al. 2004].

Traingdx

Trainrp

Trainscg

Trainlm

traingdx is back propagation method of Standard Gradient descent with momentum and adaptive learning rate (Simon Haykin,1999). The learning rate varies throughout the training process with this algorithm. In the beginning, the output from the network and the errors are computed. At the end of each iteration, new connection weights and network errors are computed again by applying the current learning rate. If the new error so calculated takes up a value exceeding the old error by more than a threshold ratio, new weights and biases are rejected. The learning rate is lowered, error is re determined, and the process is continued (Wong et al. 2007). On the other hand, if the new error shows reducing trend compared to old one, the changed weights and bias are preserved and learning rate is further improved. Using the above reiterative procedure, the best learning rate for the given problem is obtained (Matlab, 2008).

Trainrp is resilient back propagation method which removes the detrimental effects of partial derivatives. Magnitude of derivative has no impact on the process of weight update. Positive or negative sign of the derivative decides the direction of weight update (Wong, B. K. et al. 2007).

Trainscg (Scaled conjugate gradient back propagation) is a conjugate gradient algorithm. In this method, quicker convergence is achieved than other methods using the search strategy along conjugate directions (Lau, H.C.W. 2013).

4.2.2 Trainlm as best training method

Trainlm method, which is known as Levenberg–Marquard (LM) training, converges faster and also has higher stability (Yu-Hsin et al. 2009). This algorithm was found to be highly appropriate for training small and medium-sized problems, and therefore most apt in the present research work where the training data is a medium sized time series consisting of around 110 past historical sales figures.

The Levenberg–Marquardt algorithm balances the faster convergence advantage of the Gauss–Newton algorithm with stability of steepest descent method. It becomes more robust by inheriting both these properties and can converge much faster even if the error surface is more complex than quadratic condition.

The primary advantage of trainlm comes from the fact that it accomplishes a hybrid training process. The LM algorithm shifts to steepest descent when it has to pass through the area of intricate curvature, till the local curvature simplifies to a quadratic approximation. Then, it switches back to Gauss newton algorithm which rapidly converges.

On confirmation of Trainlm as the best training method, C++ program was developed for ANN model for demand forecast using this training method.

Detailed description of ANN model for the demand forecast of 2 products is furnished here for better generalization.

4.2.3 MLP Model 1: Demand Forecast for Globe Valve 10’’X 150 GTV 101

ANN models based on MLP, for demand forecast of two types of products were developed with the help of the historical sales data. Accordingly, Model 1 is for **Globe**

valve 10’’X 150 GTV 101 series and Model 2 is for Gate valve **6’’X 600 GTV 102 series.** Under each case, Demand forecast obtained from different training approaches were compared to identify the most suitable architecture which gives good generalization capability.

The scheme of training using MLP is as shown in fig 4.1. The data set was segregated into 3 parts. The first part consisting of 70% of data was utilized as training set, next 15% was used as testing data. The remaining 15% was employed for validation purpose.

Input for the neural network demand forecasting model:

Previous bimonthly sales data.

Second previous bimonthly sales data (sales of last 3rd and 4 th month)

Moving average of last 2 bimonthly sales data

Moving average of last 3 bimonthly sale data

Output of neural network is the forecasted demand for the next bimonthly sales.

The data inputs to the ANN model went through a preprocessing stage. Because of this, data got transformed into common number range (0, 0.9). This is to ensure that all the data contributed equally to the model.

Normalising function adopted was

$$x_n = \frac{(x - x_{min}) * 0.8}{(x_{max} - x_{min})} + 0.1$$

Where x_n refers to the normalised value, x_{min} is the minimum and x_{max} is the maximum value in the range (Negalye et al. 2012). Upon completion of training, post processing of output is carried out. The denormalising equation used was (Negalye et al. 2012).

$$x = \frac{(x_{max} - x_{min}) * (x_n - 0.1)}{0.8} + x_{min}$$

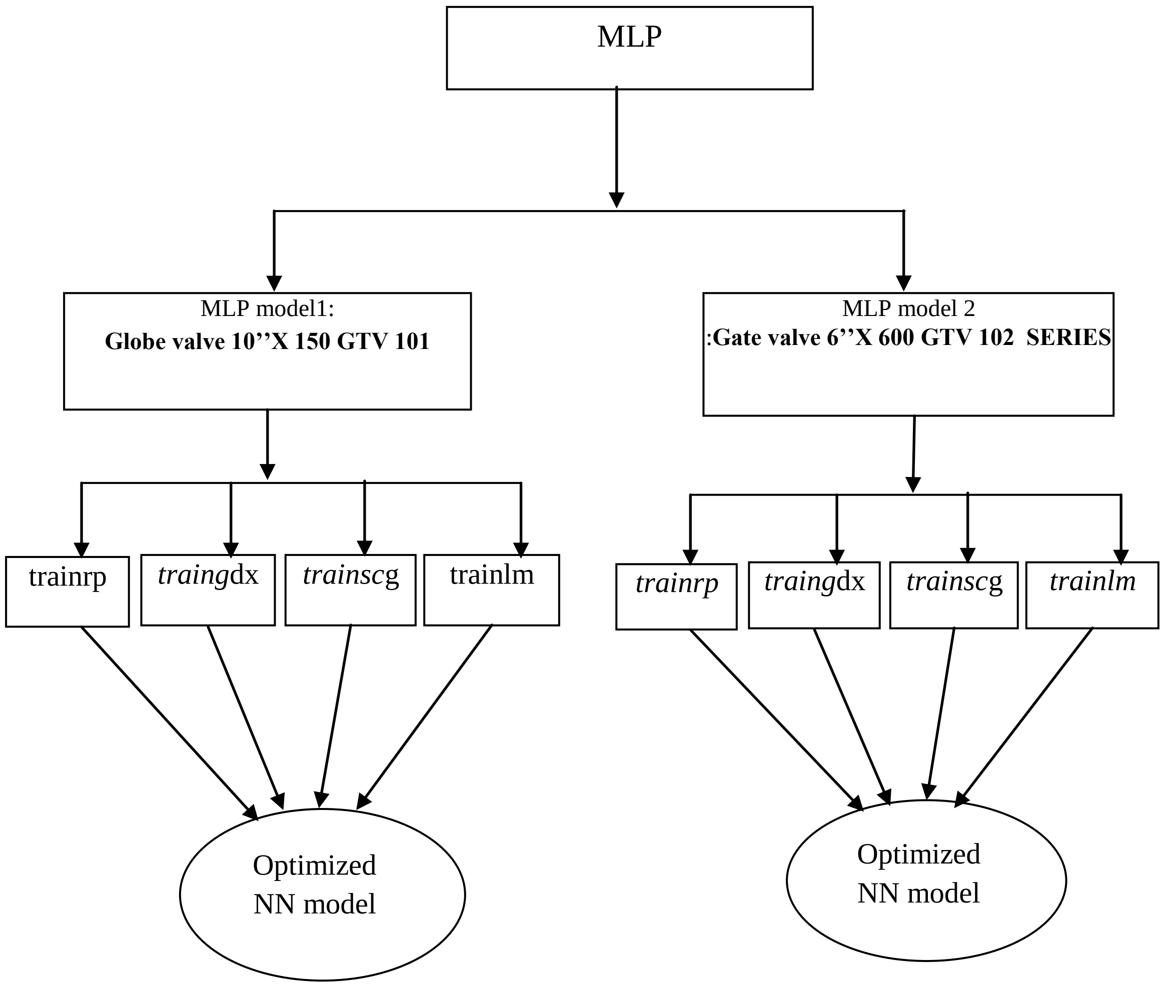


Fig 4.1 Scheme of training MLP.

In the beginning, Single hidden layer was utilized with tan sigmoid activation function. The output layer had linear activation function.

The performance of MLP is highly influenced by the number of neurons in the hidden layer. To start with, a randomly selected number of neurons in the hidden layer trained the network. Based on the MSE, the number of neurons increased or reduced to improve the performance of MLP. Lesser the MSE, better is the performance. An optimum number of neurons in hidden layer was the one for which the MSE was minimum.

The trial and error method was adopted to optimize the number of neurons in the hidden layer as there is no other standard procedure (Beale et al.2008). Table 4.1 shows the different number of nodes and the Mean Square Error obtained. Fig 4.2 shows the plot of MSE on Y- axis and number of neurons on X- axis. It is evident from the graph that MSE was 0.22 with hidden layer containing 10 neurons. MSE continuously got improved with increase in the number of neurons to reach a minimum value of 0.004 at 20 neurons. Beyond this point, there was the reversal of the trend. There was slow increase of MSE with increase in number of used neurons. This experiment established the optimum number of neurons used in the hidden layer as 20 for the further investigations. The final architecture of MLP model 1 is shown in Fig 4.3. This figure depicts inputs and outputs of the MLP model and also the number of hidden layers.

Table 4.1 Identification of Optimum No. of Nodes For Multilayer Perceptron Neural Network

No. of nodes	Mean Square Error
2	0.70
3	0.71
5	0.59
7	0.28
8	0.25
10	0.22
12	0.20
14	0.04
15	0.13
16	0.07
17	0.07
18	0.01
20	0.004
22	0.06
24	0.15
26	0.34

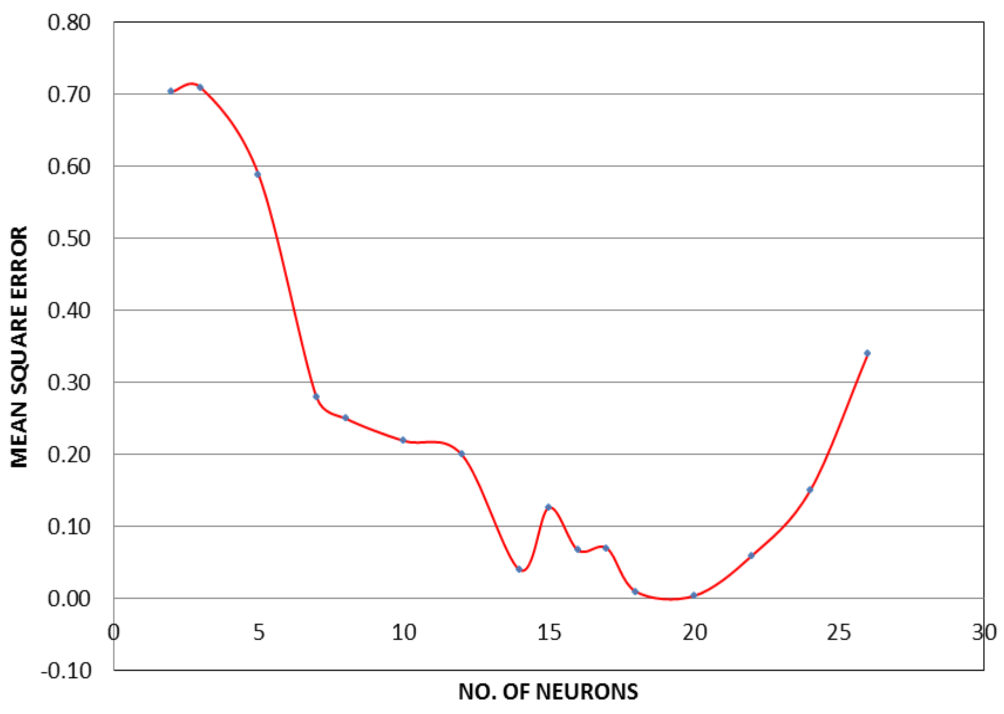


Fig 4.2 Identification of optimum number of Neurons

Four different training algorithm were used for training the MLP network. The MSE achieved was noted as in Table no. 4.2. For all the four training methods the limiting number of epoch was set at 2000. Momentum value was fixed at 0.09 and learning rate was set at 0.6 (Megalie et al. 2003). Training Mean square error goal was given a value of 0.001. Training was stopped when any one of the following three conditions were satisfied: Performance goal reach or validation error crossing test error or crossing maximum number of epochs. It can be inferred from the table 4.2 that Trainlm method gives the least MSE, and therefore best for prediction.

Further experiments were also carried out and proved that increasing the number of epochs could not reduce the MSE or improve the prediction accuracy.

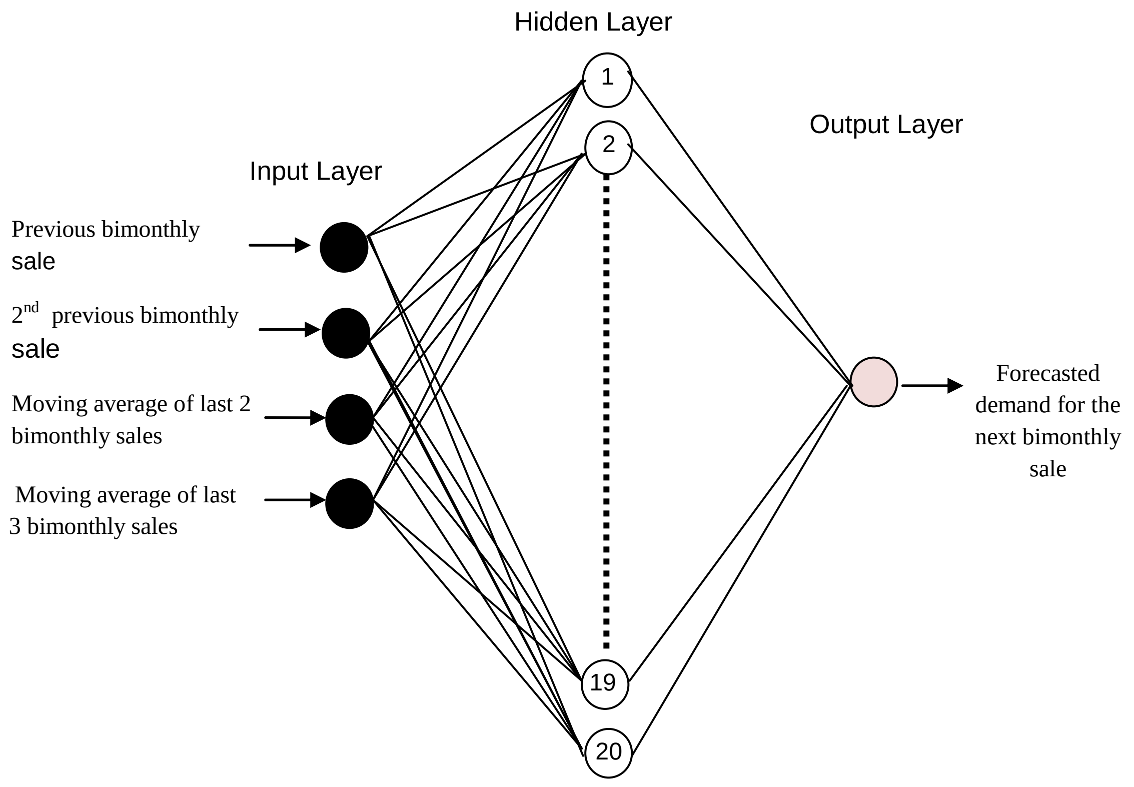

Hidden Layer
Output Layer
Input Layer
Previous bimonthly sale
2nd previous bimonthly sale
Moving average of last 2 bimonthly sales
Moving average of last 3 bimonthly sales
1
2
19
20
Forecasted demand for the next bimonthly sale

Fig 4.3 MLP model 1: Demand forecast for Globe valve 10''X 150 GTV 101

The performance of trained network was tested using Mean Absolute Percentage Error (MAPE) as the performance parameter. MAPE is defined as

$$\text{MAPE} = 1/n \sum_{t=1}^{n} \left(\frac{Y_t - F_t}{Y_t}\right) * 100$$

where Y_t is the actual demand for time period t and F_t is the forecast for the same period predicted by ANN model, n denotes the quantity of data under consideration (Tugba Efendigil et al. 2009).

Table 4.3 shows the actual values and forecasted values for the test data when Trainlm method is used for training the neural network. The absolute percentage error varies from 2.8 % to 7.8% as can be seen from the Table 4.3. Mean absolute percentage error is

4.91% with the trainlm method. Fig. 4.3 is the plot showing the demand forecast and actual sales of the corresponding time period when Trainlm method is used. It is evident that actual sales closely follow the forecasted demand as the demand predictions are highly accurate.

Table 4.4 shows the demand forecast vs actual sales for 10 X 150 GTV101 valve series employing MLP (Trainrp Method). The mean Mean absolute error is higher at 11.19%. Fig 4.4 graphically represents that there is wider difference between the forecasted demand and actual sales realized when the demand forecast is done using Trainrp method for the neural network. Table 4.5 and Fig.4.6 compares actual sales figures with the demand forecast using traingdx method. A still higher mean absolute error is recorded at 15.02%.

The actual sales figure and demand forecast predicted by Trainscg training method of neural network is shown in the Table 4.6. Fig. 4.7 represents the above comparison graphically.

Comparison of data from the tables 4.4 to 4.7 shows the MAPE value of 4.91%, 11.69%, 15.02% and 11.16%, respectively from trainlm, trainrp, traingdx and trainscg methods of neural network training. This establishes the supremacy of trainlm method which has yielded the minimum MAPE or the best prediction accuracy.

Table 4.2 Mean Square Error for different training algorithms for MLP model 1

Name of Algorithm used	**Mean Square Error**
trainrp	0.0734
traingdx	0.1140
trainscg	0.0656
trainlm	0.0040

Table 4.3 10 X 150 GTV101 Demand Forecast Vs. Actual Sales employing MLP (Trainlm Method).

Actual Sales (Qty)	Forecast Demand(Qty)	Absolute Error (%)
63	64.77	2.81
44	41.36	6.00
62	59.23	4.47
54	50.20	2.67
68	64.28	5.47
56	51.55	7.95
74	71.58	3.27
70	75.26	3.81
65	67.34	3.60
55	53.43	2.85
56	57.76	3.14
45	42.36	0.09
56	53.78	3.96

Mean Absolute Percentage Error = 4.91%

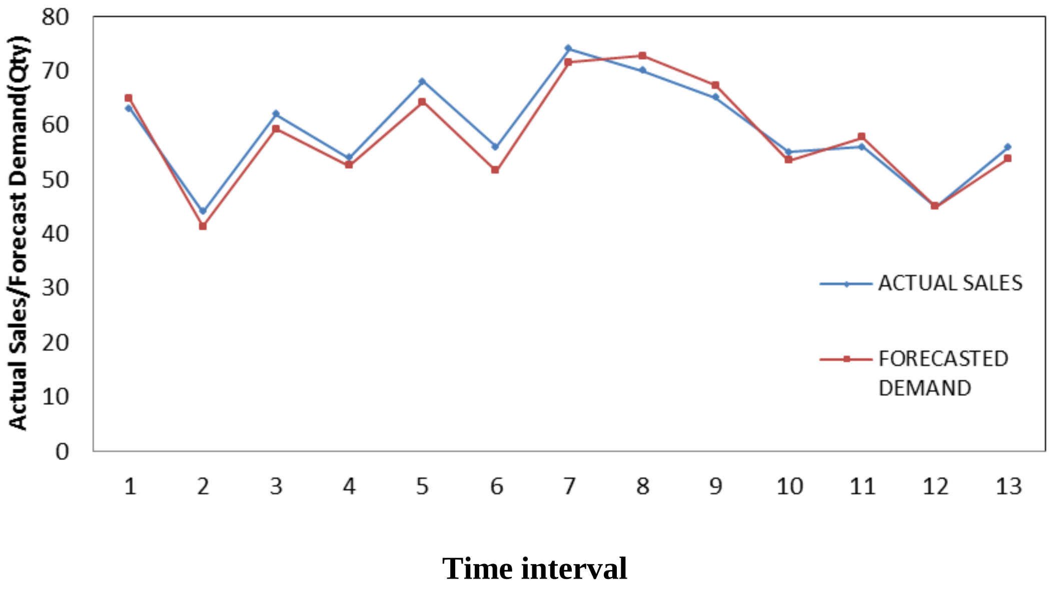


Fig.4.4 Acutal sales Vs. Demand forecast utilising MLP (Trainlm Method).

Table 4.4 10 X 150 GTV101 Demand forecast vs Actual sales employing MLP (Trainrp Method).

Actual Sales (Qty)	Forecast Demand(Qty)	Absolute Error(%)
63	63.13	0.20
44	54.69	24.29
62	66.00	6.44
54	57.68	6.82
68	64.40	5.29
56	71.20	27.15
74	58.91	20.39
70	57.66	17.64
65	55.27	14.97
55	53.67	2.41
56	51.36	8.29
45	52.43	16.52
56	56.85	1.51

Mean Absolute Percentage Error = 11.69%.

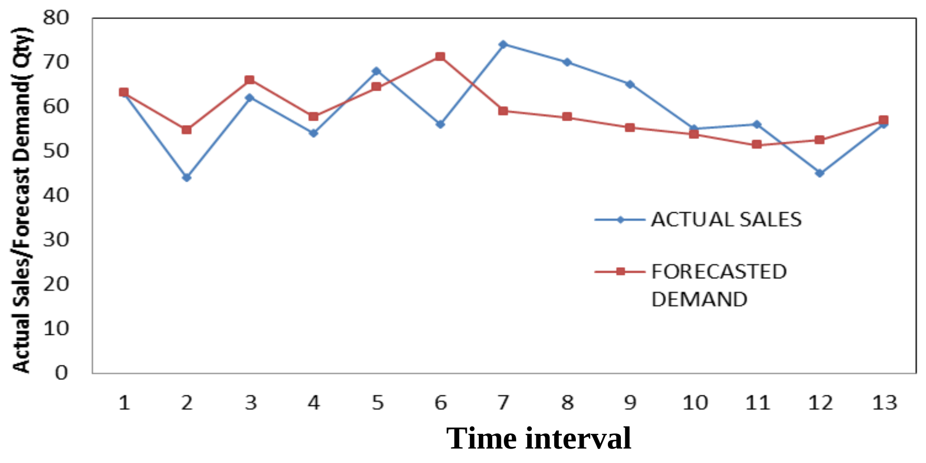


Fig 4.5 10 X 150 GTV101 Demand forecast vs Actual sales employing MLP (Trainrp method).

Table 4.5 10 X 150 GTV101 Demand Forecast Vs Actual Sales employing MLP (Traingdx Method).

Actual Sales (Qty)	Forecasted Demand(Qty)	Absolute Error (%)
63	58.03	7.89
44	60.17	36.75
62	69.39	11.92
54	62.91	16.51
68	61.33	9.81
56	63.41	13.23
74	58.64	20.76
70	57.64	17.66
65	57.05	12.24
55	54.81	0.34
56	45.99	17.87
45	56.35	25.22
56	58.82	5.03

Mean Absolute Percnt Error = 15.02%

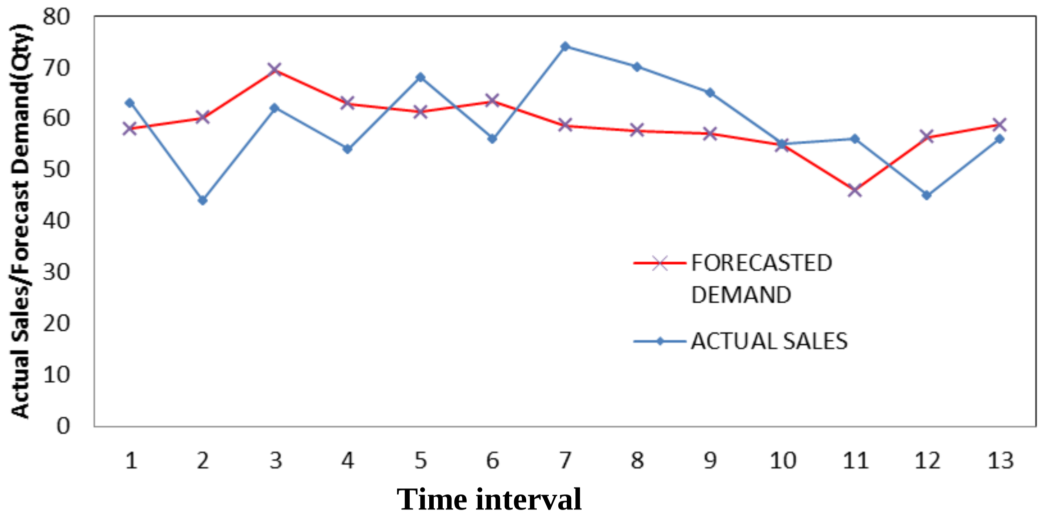


Fig. 4.6 10 X 150 GTV101 Demand Forecast Vs. Actual Sales employing MLP (Traingdx Method).

Table 4.6 Demand Forecast Vs. Actual Sales under MLP (Trainscg Method).

Actual Sales (Qty)	Forecast Demand(Qty)	Absolute Error (%)
63	63.66	1.05
44	58.00	31.82
62	50.63	18.34
54	61.37	13.65
68	71.29	4.84
56	52.21	6.77
74	53.52	27.67
70	68.59	2.01
65	61.12	5.98
55	53.85	2.10
56	53.01	5.33
45	51.59	14.65
56	62.06	10.82

Mean Absolute Error = 11.16%

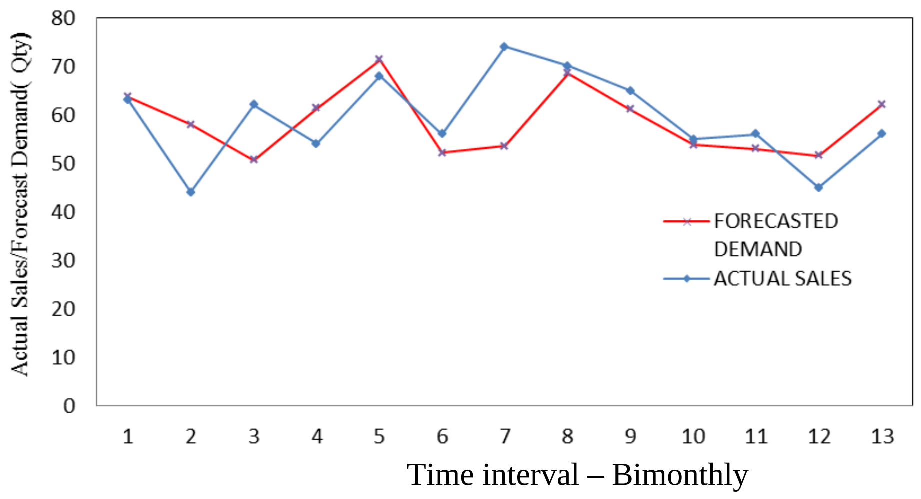


Fig 4.7 Demand Forecast Vs Actual Sales with MLP (Trainscg Method).

Table 4.7 Mean Absolute Percentage Error for MLP model 1 with different training algorithms.

Training Method	Trainrp	Traingdx	Trainscg	Trainlm
Mean Absolute Error	11.69	15.02	11.16	4.91

The same set of experiments were repeated using 2 hidden layers with 10 & 20 neurons per hidden layer. Table 4.8 lists the forecasting accuracy in terms of percentage error in prediction under different scenario with different training methods. It can be seen that the optimum architecture is 1 layer of 20 hidden neurons with trainlm training method which gives the best demand forecast accuracy at 4.91% error. It can also be observed that trainlm training method always gives better result than other training methods with the different architecture. It can also be inferred from the table that no much improvement in the forecasting accuracy is observed with increase in the number of hidden layers. This can be attributed to the fact that proper balance of model complexity and approximation ability is achieved with single hidden layer 20 neurons.

Table 4.8 Comparison of Percentage Error in forecasting using MLP with different no. of neurons And hidden layers.

Training Method	10 neuron 1 layer	20 neuron 1 layer	10 neuron 2 layer	20 neuron 2 layer
MLP-TRAINLM	7.54	4.91	7.91	5.29
MLP-TRAINRP	15.42	11.69	12.06	12.18
MLP-TRAINGDX	15.85	15.02	15.36	15.25
MLP-TRAINSCG	11.45	11.16	11.97	11.35

4.2.4 MLP Model 2: For 6’’X 600 GTV 102 Valve Series

In another study, MLP modeling was done using sales data for the 6’’X 600 GTV 102 valve series**.** Same procedure was followed for the development of the MLP model. Out of the total experimental data, 70% was used for training, 15% for validation and remaining 15% for testing purpose. Initially, the network was trained with 2 neurons in the hidden layer. The number of neurons in the hidden layer was increased and each time Mean Square Error (MSE) was recorded. The correlation of Mean Square Error with number of neurons is clearly depicted in Fig. 4.8 and table 4.9. With 2 neurons, the MSE recorded was 0.64. With the increase in the number of neurons deployed, the MSE got reduced, until an optimum number of neurons was reached. In this case, for 20 neurons MSE was 0.01 which was found to be minimum. Further increase of neurons resulted in increase of MSE. The final MLP architecture for demand forecast for 6’’X 600 GTV 102 valve series**.** shown in Fig 4.9.

Table 4.9 Identification of optimum number of Neurons for MLP model 2.

No. of nodes	Mean Square Error
2	0.64
5	0.49
8	0.43
10	0.22
15	0.07
20	0.01
25	0.25
30	0.84

Fig 4.9 clearly shows the inputs for the MLP model, the number of neurons in the hidden layer, number of hidden layers and output of the neural network model. Network was trained using *trainrp, trainscg, traingdx* and *trainlm* algorithms in the same procedure as adopted for Model1

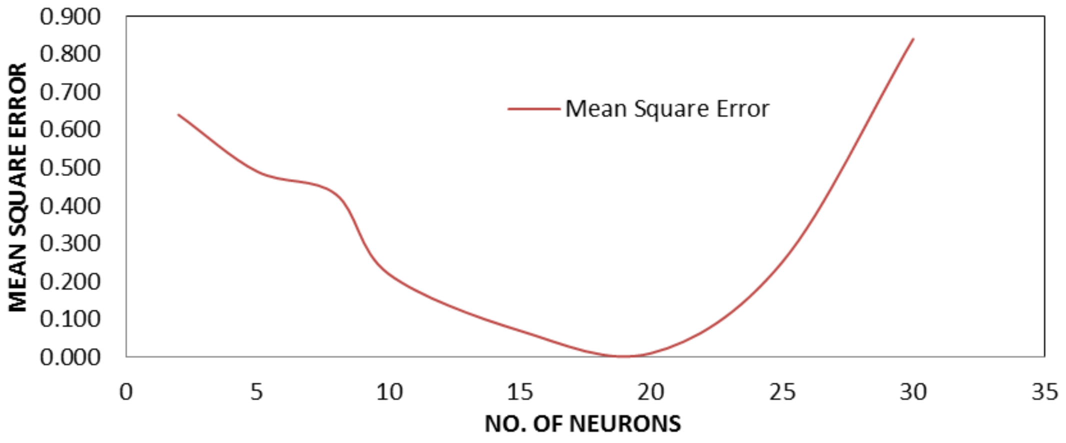


Fig 4.8 Identification of optimum number of neurons for MLP model 2.

MSE observed using each of the training algorithms is listed below in table 4.10. It shows the value of 0.784%, 0.1646%, 0.0715% and 0.0042% respectively for *trainrp*, *trainscg*, *traingdx* and *trainlm* methods of neural network training. It can be concluded from the Table 4.10 that *trainlm* outperforms other algorithms as the MSE that could be reached was the least with trainlm when compared to other algorithms.

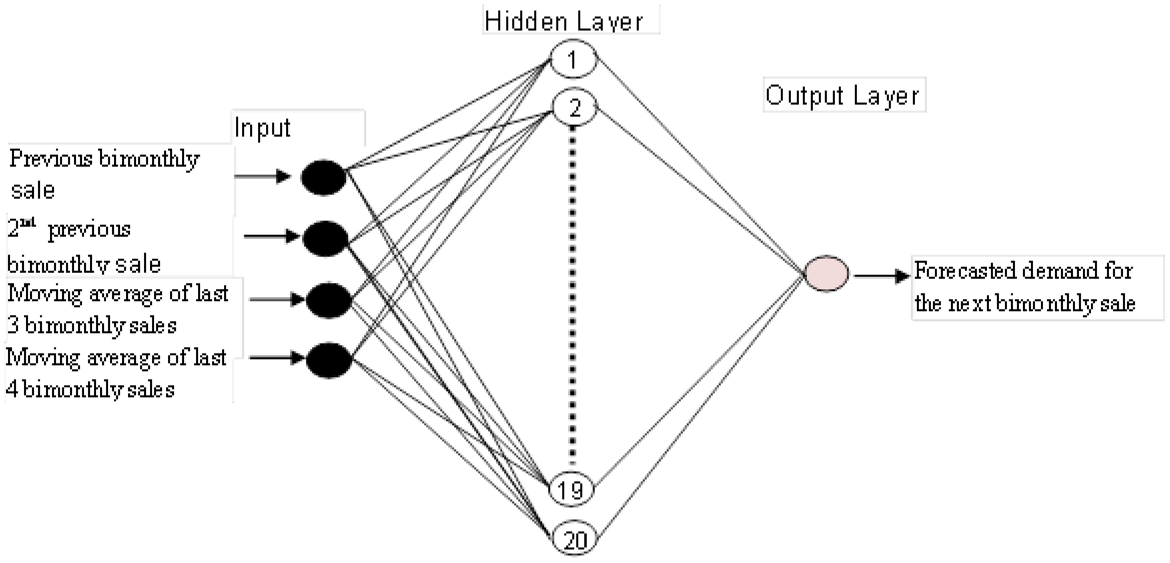

Hidden Layer
Output Layer
Input
Previous bimonthly sale
2nd previous bimonthly sale
Moving average of last 3 bimonthly sales
Moving average of last 4 bimonthly sales
1
2
19
20
Forecasted demand for the next bimonthly sale

Fig4.9 MLP model 2 : 6''X 600 GTV 102valve series.

Table 4.10 Mean square error achieved for various training algorithms-MLP model 2.

Name of Algorithm used	**Mean Square Error**
trainrp	0.0784
traingdx	0.1646
trainscg	0.0715
trainlm	0.0042

Similarly, a comparative study of the performance of the evolved MLP model for different training algorithms revealed that *trainlm* outperformed other algorithms in terms of lower MAPE or higher prediction accuracy as shown in Table 4.11. MAPE with trainlm training was as low as 5.5%, whereas it was quite high at 12.58%, 15.35% and 11.96% for trainrp, traingdx and trainscg methods respectively.

Table 4.11 Mean Absolute Error for MLP model 2 with different training algorithms.

Training Method	Trainrp	Traingdx	Trainscg	Trainlm
Mean Absolute Error (%)	12.58	15.35	11.96	5.5

MLP model 2 was also tried by varying the number of layers, number of neurons in each layer and training methods. Table 4.12 shows the prediction accuracy achieved in each case. It lists the MAPE achieved for each training method in different cases of adopting 10 and 20 neurons with one and two hidden layers. Increasing the number of neurons from 10 to 20 has improved considerably prediction accuracy with all training methods. Substantial improvement was observed in case of trainlm learning. The MAPE got improved from 8.56% to 5.5 % with the increase in the number of neurons from 10 to 20. Anyway, altering the number of layers from 1 to 2 did not improve the result in any of training methods. It is evident from the table that best MLP model 2 architecture is one neuron layer with 20 neurons and trained with trainlm method.

Table 4.12 Comparison Of Percentage Error in Forecasting Using MLP with Different No. of Neurons And Hidden Layers.

Training Method	10 neuron 1 layer	20 neuron 1 layer	10 neuron 2 layer	20 neuron 2 layer
MLP-TRAINLM	8.56	5.5	7.90	6.10
MLP-TRAINRP	15.65	12.58	12.56	12.30
MLP-TRAINGDX	16.15	15.35	15.76	16.02
MLP-TRAINSCG	11.88	11.96	12.55	12.15

4.2.5 Summary of results.

In this section, MLP NN model has been investigated for predicting the bimonthly demand of 10 X 150 GTV101valve series and 6''X 600 GTV 102valve series called as MLP model 1 and MLP model 2.

Learning characteristics of MLP in terms of number of neurons in the hidden layer, has been studied for faster convergence and desired level of accuracy. Table 4.1 lists the no. of nodes used for the hidden layer of MLP neural network and the resulting mean square error keeping other parameters constant using the TRAINLM method. As the number of nodes increases, the resulting mean square error decreases initially and after an optimum value, the error starts increasing. At the optimum number of neurons, that is 20, the error is minimum 0.01. The same is plotted in the graph in Fig. 4.2 taking number of neurons on X-axis and Mean square error on Y-axis. More neurons than the optimum result in overfitting. This means the ANN would overestimate the target problem complexity. It reduces the capability of the network to generalize, thereby adversely affecting the forecasting accuracy. Hence for the neural network, to effectively carry out the function approximation, it is highly essential that optimum number of neurons should be employed. This is illustrated in table 4.1 and Fig 4.2.

MLP has fixed architecture, where the number of hidden neurons was established by trial and error method. This is very time consuming.

Different back propagation training algorithms namely *trainlm, traingdx, trainrp*, and *trainscg* algorithms were used to train the network.

Performance of the algorithms or the prediction accuracy was compared in terms of MAPE. *Trainlm* algorithm performed better than other algorithms with an average MAPE of less than 5%. Results indicated that MLP can be effectively applied for demand forecast, when sufficient historical data is available.

4.3 RADIAL BASIS FUNCTION NEURAL NETWORKS

This section presents a detailed analysis of use of radial basis function neural networks for demand forecast. As already described in the section 3.6, RBF network is a feed forward neural network with only one hidden layer. Two parameters center x_j and width σ_j characterize the node of the hidden layer. Two different methods were adopted to fix the centers of RBF and two methods for width $\sigma_{j.,}$ and their learning characteristics were analyzed. Performance of RBF neural networks have been compared with MLP for prediction. The scheme of training using MLP is as shown in fig 4.10.

In the present work, RBFNN models have been developed for demand forecast of 10''X 150 GTV 101valve series & 6''X 600 GTV 102 valve series represented as RBF model 1 and RBF model 2, respectively. Center initialization and selection is achieved using two learning strategies.

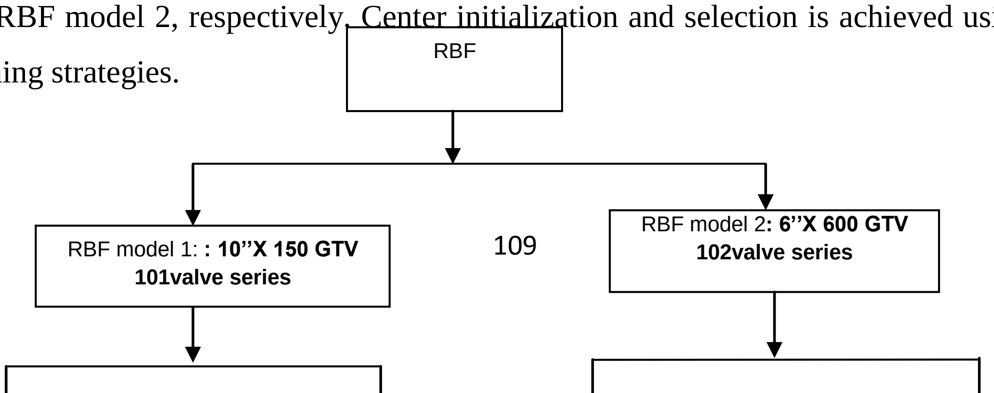

RBF
RBF model 1: : **10''X 150 GTV 101valve series**
RBF model 2**: 6''X 600 GTV 102valve series**

Fig. 4.10 Scheme of RBF network training.

Fixed centers selected at random

Self organized selection of centers based on Clustering algorithms like fuzzy C-means (FCM)

Similarly, two strategies have been used for width selection.

Fixed width

Variable width computed based on P-nearest neighbor heuristic

Demand forecast is carried out using each of the above learning strategies and prediction accuracy is determined for both models.

4.3.1 RBF Model 1: Demand forecast of 10’’X 150 GTV 101 Valve series

The data set was segregated into 3 parts. The first part consisting of 70% of data was utilized as training set, next 15% was used as testing data. The remaining 15% was employed for validation purpose.

Input for the neural network demand forecasting model:

1. Previous bimonthly sales data.
2. Second previous bimonthly sale data (sales of last 3rd and 4th month).
3. Moving average of last 3 bimonthly sales data.
4. Moving average of last 4 bimonthly sale data.

Next bimonthly demand forecast is obtained as the output of neural network. C++ programming platform has been explored for evolving the code for RBF network implementation using different learning strategies. Similar to MLP method, data preprocessing is carried out so that the input data are in the range of (0, 0.9)

4.3.1.1 RBF training and testing with centers selected randomly

The algorithm used for this training has been explained in chapter 3. Optimum value of parameters learning rate η =0.85 and momentum α=0.05 was selected and used throughout experiments and evaluations.

With the centers selected randomly two studies were done:

With variable width of RBF units determined using P-nearest neighbor heuristic
With fixed and equal width for RBF units.
To study the convergence behavior, the maximum number of cycles was fixed at 2000 epochs. The program was run and network error was tabulated for different number of centers. The network goal error was taken as 0.001. It is evident from the Table 4.13 that there was a decrease in error and reached a minimum of 0.001 for the number of centers 110. After 110, the error starts increasing again. Therefore, 110 was chosen as optimum number of centers. The RBF structure is illustrated in the Fig. 4.13. It shows the different input parameters for RBF model, the number of random centers and output of the model which is the demand forecast for next period.

With the number of centers as 110, the network error is noted for different number of epochs. The Fig 4.11 shows the error vs number of epochs response. As the number of epochs increases, the error in forecast decreases. It can be inferred from the Fig. 4.11 that the optimum number of epochs is at around 600. Increasing the number of epochs above 600 is not required for obtaining the best prediction accuracy.

Table 4.13 Mean square Error and number of RBF units- co-relation study for the RBF model 1(Centers selected randomly).

Number of RBF centers	50	70	80	100	110	120
MSE (%)	0.001965	0.001432	0.001298	0.00119	0.001000	0.00120

Table 4.14 shows the performance of RBF model 1 with variable width of RBF units and centers chosen randomly. It lists the actual sales, forecasted demand and prediction accuracy in terms of error in prediction. The mean absolute percentage error is recorded as 4.81%. Fig. 4.12 graphically illustrates the error in prediction by trending actual sales and forecasted demand. X-axis represents the time interval.

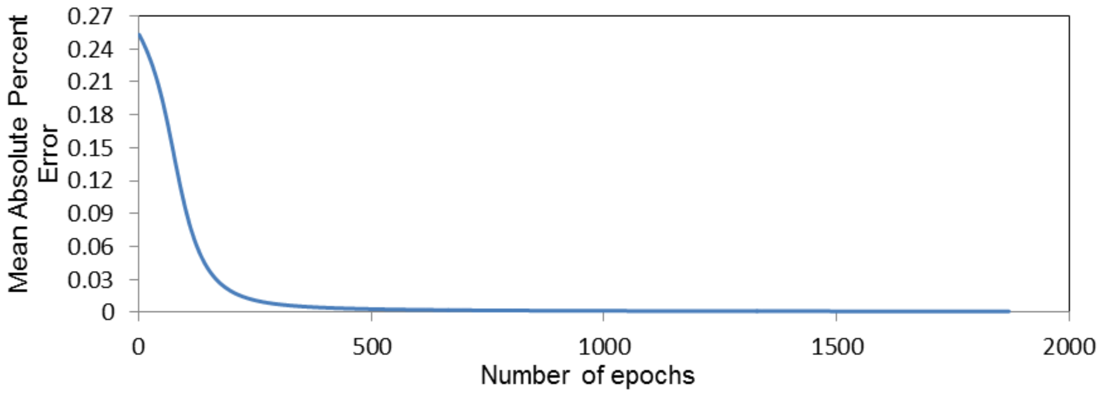


Fig 4.11. Variation of error with number of epochs for Random choice of centers for RBF model 1. (RBF centers=110).

Table 4.14 Demand forecast for 10 X 150 GTV101 Valve Series -Performance of RBF model 1 (variable width and center chosen randomly).

Actual Sales (Qty)	Demand Forecast (Qty)	Absolute Error(%)
63	63.50	0.79
44	42.00	4.55
62	59.00	4.84
54	51.00	5.56
68	63.20	7.06
56	52.50	6.25
74	70.60	4.59
70	74.80	6.86
65	67.20	3.38
55	53.10	3.45
56	58.40	4.29
45	42.60	5.33
56	52.90	5.54

Mean absolute Error =4.81%

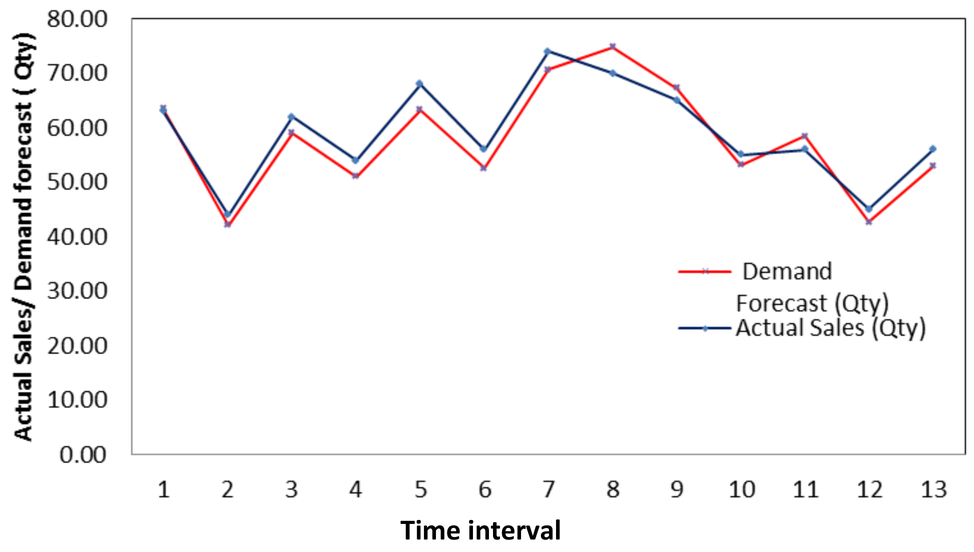


Fig. 4.12 Demand forecast for 10 X 150 GTV101 Valve Series- Performance of RBF model 1 (variable width and random choice of center).

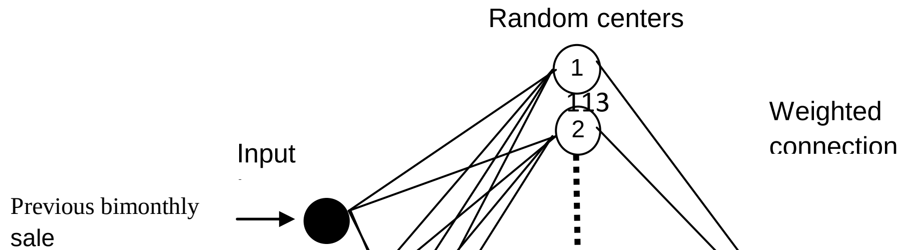


Fig 4.13 RBF model 1: Demand forecast for Globe valve 10''X 150 GTV 101.

Demand for the product is shown on Y axis. Two curves are drawn to represent the forecasted demand and actual sales. Since the error was very less at around 4.81%, the actual sales figure curve closely follows the forecasted demand curve as obvious from the Fig. 4.12.

In second part of the study with random centers, widths are not allowed to vary during the study. Keeping the width constant for all radial basis function units, network training carried out for different value of fixed widths. The optimum number of centers 110 is used in this study. Network error variation with the different values of fixed widths is captured in Table 4.15. Table clearly points out to the fact that as the width increases, the error decreases till an optimum value of width is reached. After this optimum value of width, any further increase in width will cause increase in error. The optimum value of width is 0.08 and corresponding error is 0.001.

Table 4.15 Variation of MSE with widths for RBF model 1 (Random selection of center).

RBF units Width	0.01	0.03	0.05	0.07	0.10	0.15
Mean Square Error	0.0014	0.0013	0.0010	0.0011	0.007	0.2970

The performance of the network for this fixed width is shown in table 4.16 and Fig. 4.14. Table 4.16 lists the forecasted demand and corresponding actual sales . From this data, the prediction error is computed. The mean Absolute percentage error in this case is 4.73%. The results show that MAPE decreased slightly when compared to the use of P-heuristic method with a corresponding improvement in prediction accuracy.

4.3.1.2 RBF training and testing with centers chosen utilizing fuzzy c means algorithm.

This methodology makes use of clustering algorithm namely Fuzzy C means for the self-organized selection of centers (Hadjahmadi et al. 2008).

Table 4.16 Demand Forecast for 10 X 150 GTV101 Valve Series-Performance of RBF model 1 (fixed width and random selection of centers).

Actual Sales (Qty)	Demand Forecast(Qty)	Absolute Error(%)
63	63.4	0.63
44	42.3	3.86
62	59.4	4.19
54	51.5	4.63
68	63.6	6.47
56	52.2	6.79
74	70.7	4.46
70	74.8	6.86
65	68	4.62
55	52.6	4.36
56	58.1	3.75
45	42.7	5.11
56	52.8	5.71

Mean Absolute Percentage Error = 4.73%

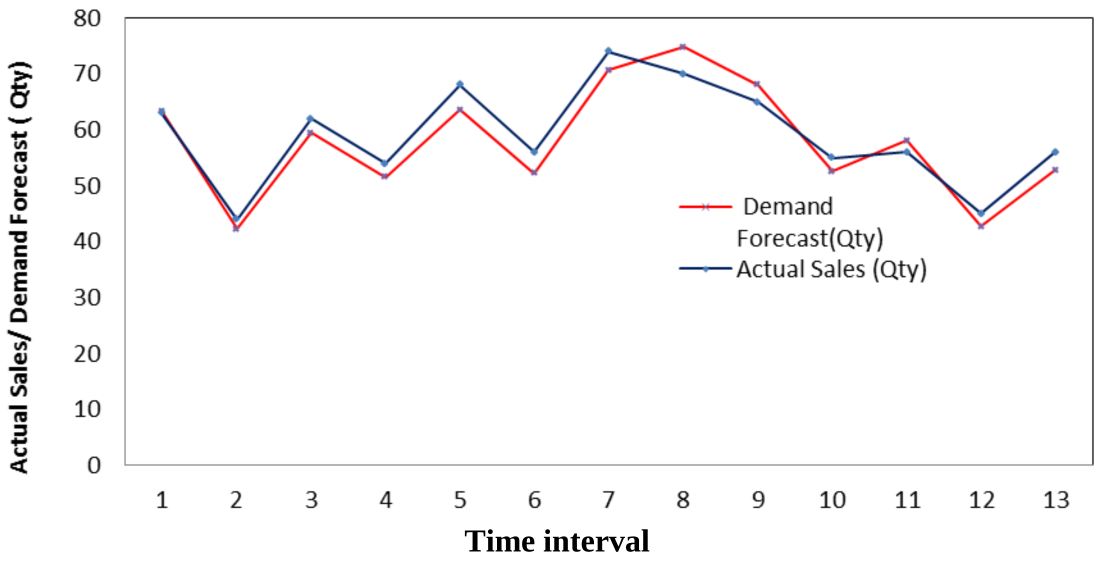


Fig: 4.14 Demand Forecast for 10 X 150 GTV101 Valve Series- Performance of RBF model 1 (fixed width and random selection of centers).

Optimum value of parameters η =0.85 and α=0.05 was selected and used through out experiments and evaluations. With the centers selected using Fuzzy C means, two studies were done:

1. With variable width of RBF units established employing P-nearest neighbor heuristic.

2. With fixed and equal width for RBF units

Table 4.17. Correlation study of Mean Square Error with number of RBF units for RBF model 1 (Centers selected using FCM).

Number of centers	40	50	60	70	80	90	100
MSE (x 10^{-3})	1.89	1.41	1.278	1.001	1.000	1.025	1.296

The training of the fuzzy c-means algorithm has been executed with various number of centers, and the final optimal number of centers obtained after convergence have been used to train the RBF network. Network was trained initially by using P-nearest neighbor heuristic width (Krishna et al. 2008). Table 4.17 illustrates the response of the network for different number of centers, with the variation of MSE. With the increase in number of centers, the error got improved till it reached the lowest at 0.001. After that increasing the number of centers had detrimental effect for network learning and error started raising. The lowest error corresponds to the number of centers 80, which is best suited for the purpose.

Fig. 4.15 depicts the final RBF structure with optimum number of centers. The figure portrays the input parameters, the hidden layer configuration with the centers selected by FCM and output from RBF neural network. Table 4.18 and Fig 4.16 show output of the demand forecast obtained using RBF network. Fuzzy C means is used to select the centers for training this network. Table 4.18 shows the forecasted demand and actual sales and prediction accuracy in terms of MAPE. It can be observed that MAPE and there by the prediction accuracy slightly improved in comparison with the fixed center selection method.

Fig 4.15 RBF model 1 (Centers selected using FCM algorithm).

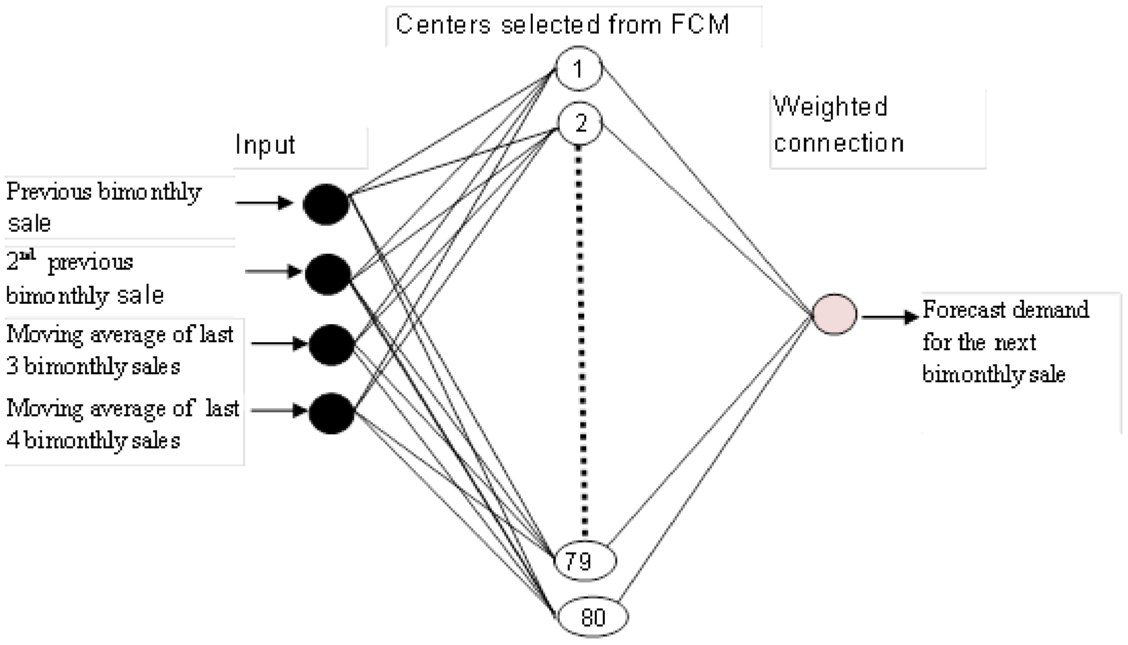

Centers selected from FCM
1
2
79
80
Input
Weighted connection
Previous bimonthly sale
2nd previous bimonthly sale
Moving average of last 3 bimonthly sales
Moving average of last 4 bimonthly sales
Forecast demand for the next bimonthly sale

Table 4.18 Demand Forecast for 10 X 150 GTV101 Valve Series- Performance of RBF model 1 (variable width and Centers selection based on FCM).

Actual Sales (Qty)	Demand Forecast(Qty)	Absolute Error(%)
63	63.4	0.63
44	42.5	3.41
62	59.7	3.71
54	51.6	4.44
68	63.6	6.47
56	52.2	6.79
74	70.9	4.19
70	74.5	6.43
65	67.7	4.15
55	52.6	4.36
56	58.4	4.29
45	43.2	4.00
56	52.8	5.71

MAPE= 4.51%

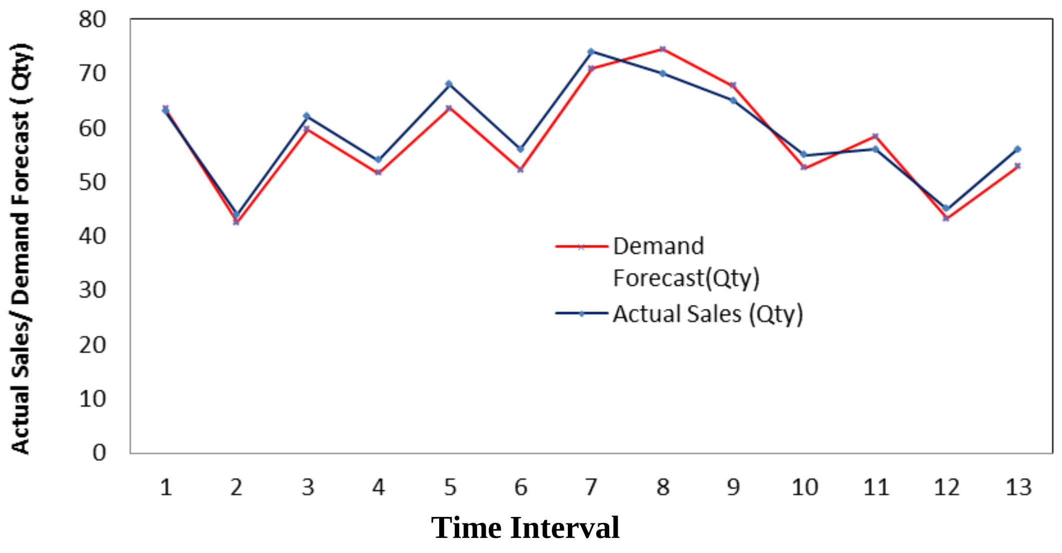


Fig. 4.16 Demand Forecast for 10 X 150 GTV101 Valve Series Performance of RBF model 1 (Variable width and Centers selection based on FCM).

When the Fuzzy C means algorithm was used, the number of centers required for optimum performance of the network got considerably reduced compared with the use of centers selected randomly. This has been attributed to the use of clustering while selecting centers which optimizes the number of centers.

In the second part of study of RBF units with self organized centers, network training was carried out using different fixed width values. The response of network for the different width values in terms of variation of MSE is shown in Table 4.19. This shows that for a width of 0.06, the MSE was found to be minimum. Using a width of 0.06 and number of centers 80, the network training was carried out and the network performance is depicted in Table 4.20 in terms of MAPE. The performance of the RBF network using fixed width values for RBF units was found to be better than with varying widths, thereby improving the prediction accuracy marginally. Fig 4.17 shows the demand forecast vs actual sales for RBF model 1 (Centers selected using FCM, Fixed width).

Table 4.19 Correlation of network width with MSE for RBF model 1. (Centers selection based on FCM).

Value of Width	0.02	0.04	0.06	0.08	0.10	0.12
Mean Square Error (x10^{-3})	2.667	1.194	1.000	1.011	2.53	3.12

4.3.1.3 Comparison of Forecast Accuracy for Different RBF Architecture

Table 4.21 lists the MAPE in the demand forecast for RBF model 1 with the different RBF architecture. Fig. 4.18 graphically illustrates the same with the bar chart. It summarises the results obtained for the demand forecast using RBF model 1 with random center and Fuzzy C means center.

Table 4.20 Demand Forecast for 10 X 150 GTV101 Valve Series-Performance of RBF model 1 (Fixed width and Centers selection based on FCM).

Actual Sales (Qty)	Demand Forecast(Qty)	Absolute Error(%)
63	63.4	0.63
44	42.7	2.95
62	59.7	3.71
54	51.5	4.63
68	63.6	6.47
56	53.2	5.00
74	70.6	4.59
70	74.1	5.86
65	67.7	4.15
55	52.8	4.00
56	58.8	5.00
45	43.2	4.00
56	52.7	5.89

MAPE=4.38%

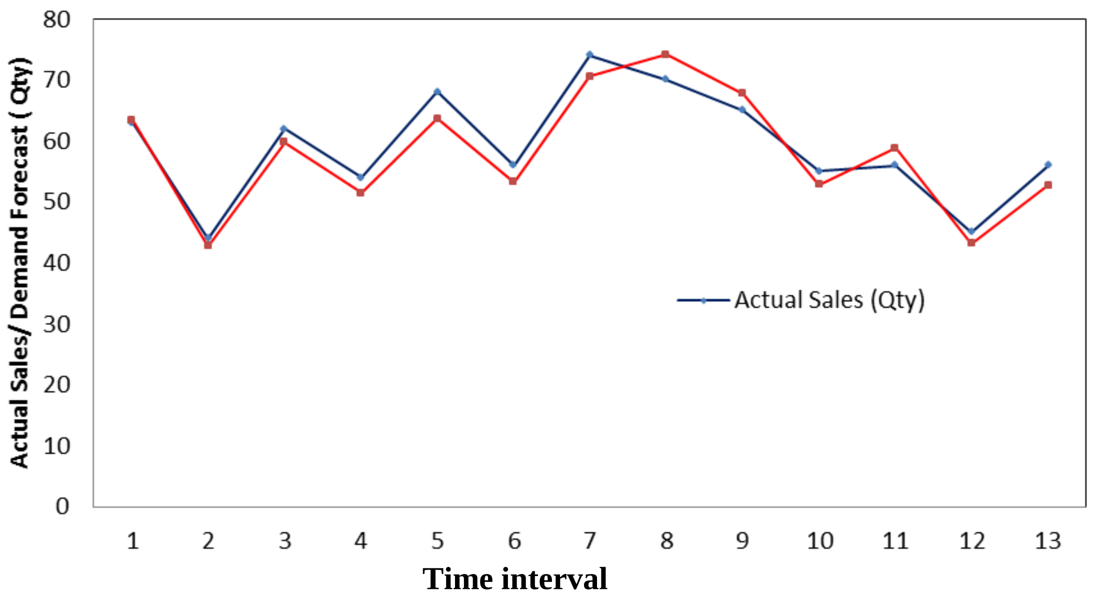


Fig. 4.17 Demand Forecast for 10 X 150 GTV101 Valve Series- Performance of RBF model 1 (Fixed width and Centers selection based on FCM).

Both fixed and variable width are considered in each case. It can be inferred that the architecture with the self organized centers selected using Fuzzy C means and fixed width has got highest prediction accuracy, even if the improvement over other methods is marginal.

Table 4.21 RBF model 1 : Demand Forecast of 10’’X 150 GTV 101 Valve Series-Comparison of Forecast Accuracy for Different RBF Architecture.

RBF Architecture	Mean Absolute Percent Error
Random center , variable width	4.81%
Random center , Fixed width	4.73%
FCM Center- variable width	4.50%
FCM Center- Fixed width	4.38%

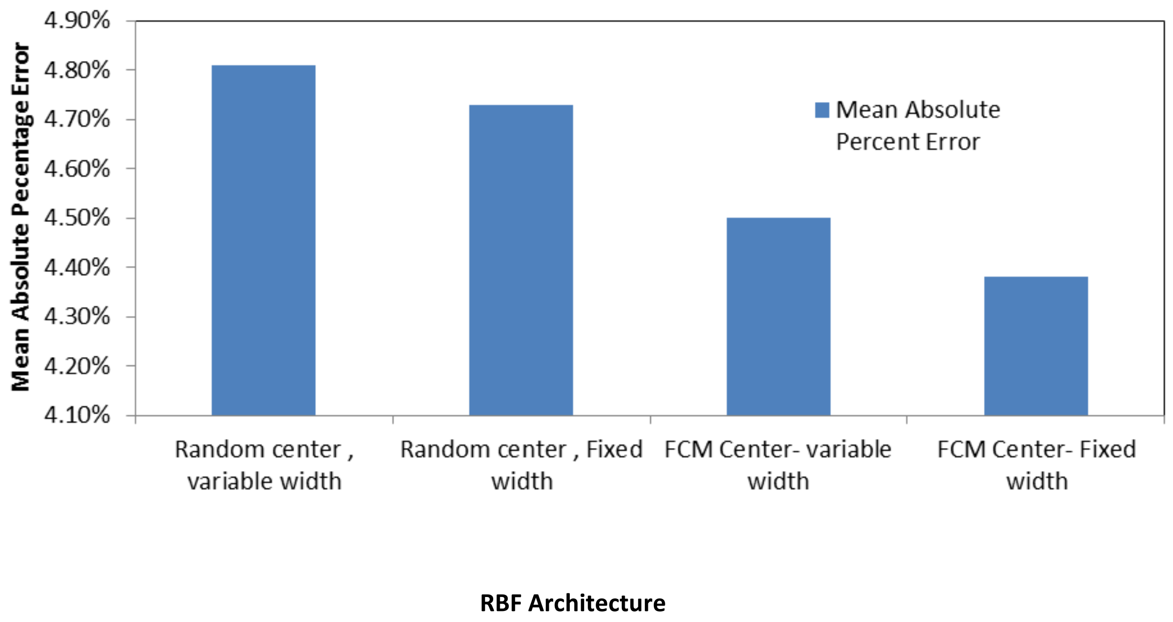


Fig: 4.18 RBF model 1 :Demand Forecast of 10’’X 150 GTV 101 Valve Series-Comparison of Forecast Accuracy for Different RBF Architecture.

4.3.2 RBF MODEL 2: DEMAND FORECAST FOR 6''X600 GTV 102 VALVE SERIES

Similar to Model no. 1, the inputs are taken from past historical sales data of the 6''X 600 GTV 102 VALVE SERIES. Same input and output configuration is followed. Same training strategy as shown in Fig 4.10 is adopted.

4.3.2.1 RBF training and testing with RBF centers selected randomly

RBF centers were selected randomly and network training was carried out employing variable width values. The variation of network error with the number of centers is shown in Table 4.22. Train and error method for obtaining the optimum number of centers was started with 75 which gave MSE of 0.00393. Error is reduced with the increase in the number of centers and the optimum number of centers obtained was 115. The correlation of Mean Square error that was achieved with the limiting number of epochs during training the network with 115 RBF units is shown in Fig 4.19. The MAPE or accuracy of prediction improved to 5%.

Table 4.22. Correlation of Mean Square Error with number of RBF units for RBF model-2 (Centers selected randomly).

Number of centers	75	90	100	105	110	115	120
MSE	0.00393	0.00373	0.0024	0.00154	0.00106	0.001000	0.00107

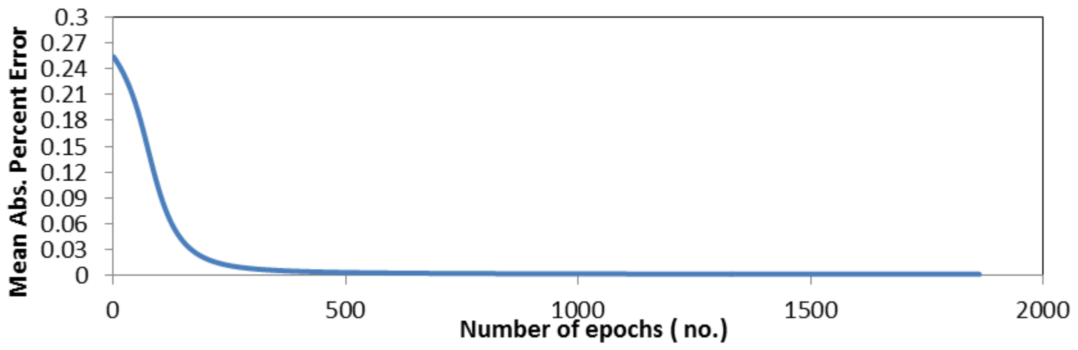


Fig 4.19 Correlation of error with number of epochs with centers selected randomly for RBF model 2 (RBF centers =115).

In next phase of study for RBF model no. 2 with randomly selected centers, network training was carried out employing different fixed value widths. For various value of widths the network error was computed and captured as shown in Table 4.23. From the table it can be inferred that for a width of 0.05, the error tapered off to a minimum value of 0.001. The network training was executed with optimum number of centers established from the random selection method. A fixed value of 0.05 was taken for the width. For RBF model 2, demand forecast of **6''X 600 GTV 102 VALVE SERIES** forecasting accuracy in terms of MAPE marginally improved to 4.95% with RBF network architecture using random selection of centers and fixed optimum width as seen from Table 4.26.

Table 4.23. Correlation of Mean Square Error and widths for RBF model 2 (Centers selected randomly).

Value of Width	0.02	0.03	0.05	0.06	0.10	0.12
Mean Square Error (x10^{-3})	2.0	1.31	1.00	1.08	8	21.34

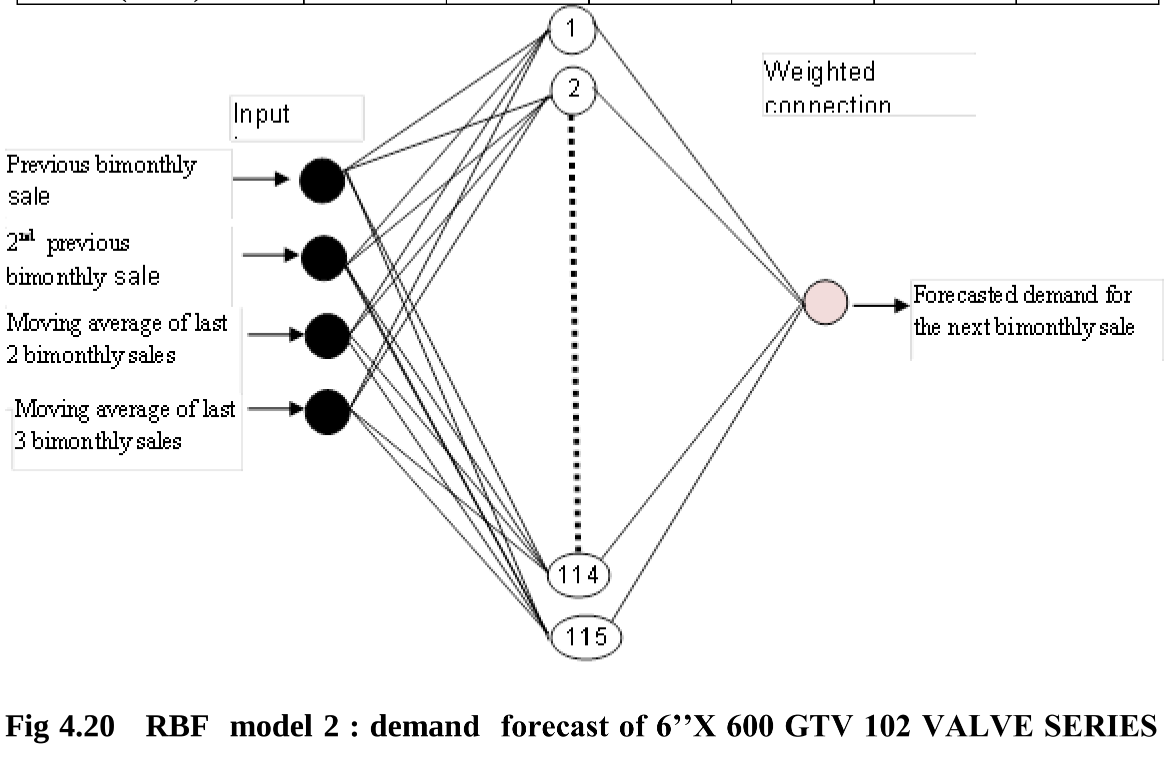

Input
Weighted connection
Previous bimonthly sale
2nd previous bimonthly sale
Moving average of last 2 bimonthly sales
Moving average of last 3 bimonthly sales
1
2
114
115
Forecasted demand for the next bimonthly sale

Fig 4.20 RBF model 2 : demand forecast of 6''X 600 GTV 102 VALVE SERIES (Centers selected randomly).

4.3.2.2. RBF training and testing with centers selection based on Fuzzy C means algorithm

RBF network architecture was decided based on center selection utilizing the Fuzzy C means. Value of width was established based on P- nearest neighborhood heuristic. It was noted that the network error was higher when the number of centers was less. With the increase in the number of centers, MSE decreased and optimum value was obtained for 95. (Table 4.24). Fig 4.20 shows the complete structure for RBF model 2 with above architecture.

On comparison with the random selection method, there was considerable decrease in the number of centers for optimum network performance. It was observed that there was a slight improvement in the performance demand forecast and MAPE was 4.82%, as shown in Table 4.26

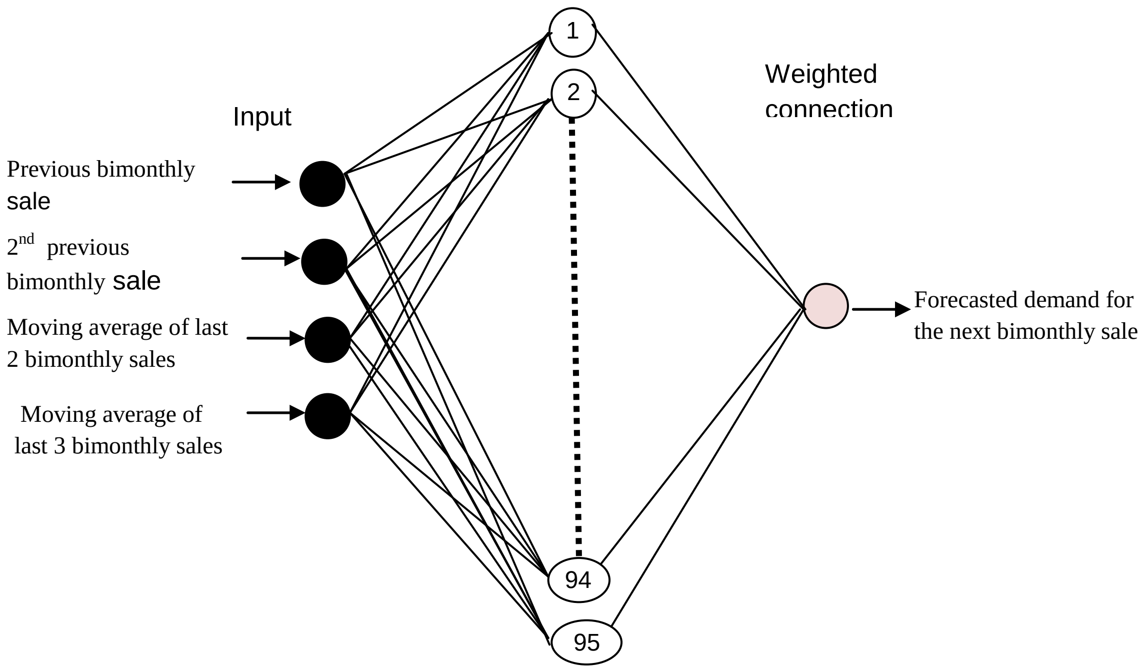
Table 4.24 Correlation of MSE and of RBF centers for RBF model 2 (Centers selected using FCM).

Number of centers	70	80	90	95	100	105	115
MSE	0.0017	0.0014	0.0012	0.001	0.00102	0.00103	0.00129

Further the network was trained using fixed width values for the optimized number of RBF centers. The error value decreased as width value increased and the optimum value reached corresponding to a width of 0.07 (Table 4.25). The performance of the network was analyzed for this value of width. It can be inferred from the Table 4.26, that Mean Square error slightly decreased to 4.65% and prediction accuracy improved slightly. Hence the prediction performance obtained employing fuzzy C means algorithm with fixed width values was better than with RBF centers selected using random selection of centers**.**

Table 4.25 Correlation of MSE and width value for RBF model 2 (Center selection based on FCM).

Value of width	0.01	0.03	0.07	0.10	0.13	0.16
Mean Square error ($X10^{-3}$)	22.1	2.13	1.00	1.34	3.55	5.45



Input
Previous bimonthly sale
2nd previous bimonthly sale
Moving average of last 2 bimonthly sales
Moving average of last 3 bimonthly sales
1
2
94
95
Weighted connection
Forecasted demand for the next bimonthly sale

Fig 4.21 RBF model 2 : demand forecast of 6''X 600 GTV 102 VALVE SERIES (Centers selected with FCM).

4.3.2.3 Comparison of Forecast Accuracy for Different RBF Architecture for RBF model 2

Table 4.26 lists the MAPE in the demand forecast for RBF model 2 with the different RBF architecture. Fig. 4.22 graphically illustrates the same with the bar chart for quick visualization and better comprehension. The random center RBF architecture gives a MAPE of 5.0% and 4.95% with variable width and fixed width respectively.

Table 4.26 RBF model 2: Demand Forecast of 6''X 600 GTV 102 Valve Series-Comparison of Forecast Accuracy for Different RBF Architecture.

RBF Architecture	Mean Absolute Percent Error
Random center , variable width	5.0%
Random center , Fixed width	4.95%
FCM Center- variable width	4.82%
FCM Center- Fixed width	4.65%

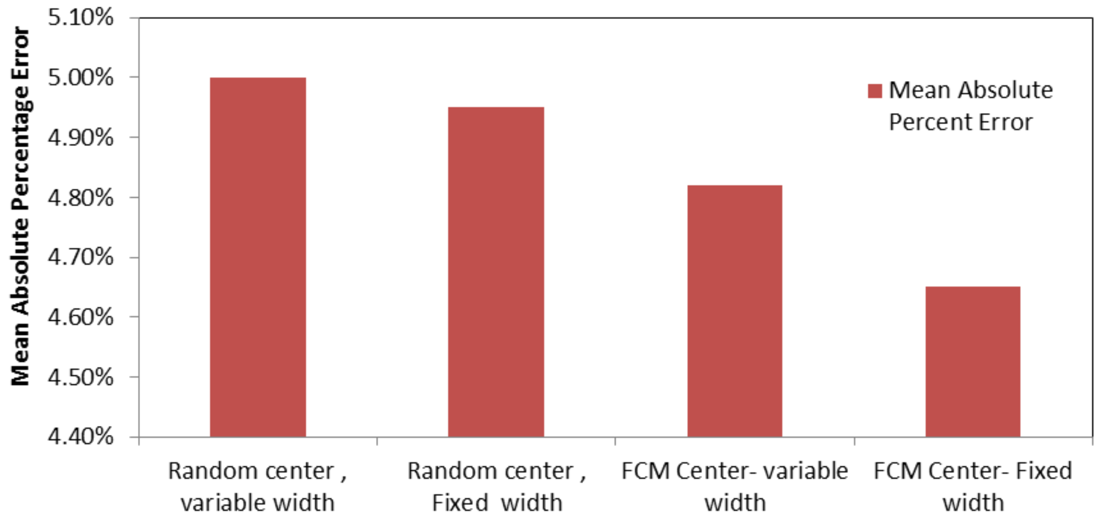


Fig 4.22 RBF model 2 : Demand forecast of 6''X 600 GTV 102 Valve Series-Comparison of Forecast Accuracy for Different RBF Architecture.

It can be inferred that the architecture with the self-organized centers selected using Fuzzy C means and fixed width has got the highest prediction accuracy, even if the improvement over other methods is marginal. Both in the RBF model 1 & 2, the performance of the different RBF architecture has been consistent.

4.3.3 SUMMARY OF RESULTS

1. Two approaches were adopted for center selection for investigating the effectiveness of RBF neural networks. One of the methodologies was random choice of centers, other was use of FCM. The limitation of random selection of centers was the difficulty in determining the right number of RBF units using trial and error method On the other hand, Use of FCM algorithm facilitated the determination and location of the optimum number of centers for required level of performance. FCM algorithm outperformed random selection of centers with respect to prediction accuracy. Investigations also revealed that results acquired employing the fixed widths for the RBF centers were more accurate than results from the variable width architecture.

2. Table 4.27 lists out the different architectures used for neural network demand prediction. It includes MLP and variations of RBF network. For every type of network architecture, MAPE realized for both demand forecast models is recorded. Fig. 4.23 presents the above data in the form of bar chart. Effectively, Table 4.27 and Fig. 4.23 show the comparative evaluation of RBF networks using two different methods of center selection with multilayer perceptron for model 1 and 2.

The evaluation has been done on the basis of the minimum Mean Absolute Percentage Error that the different algorithm can achieve in terms of prediction accuracy. This is a measure of how accurately the algorithm can predict the future demand based on the previous data.

Table 4.27: Comparison of forecasting accuracy for MLP and different RBF network for model 1 & 2.

Network	MAPE for Demand Forecast model-1	MAPE for Demand Forecast model-2
MLP	4.91%	5.3%
Random center - variable width	4.81%	5.0%
Random center - Fixed width	4.73%	4.95%
FCM Center- variable width	4.50%	4.82%
FCM Center- Fixed width	4.38%	4.65%

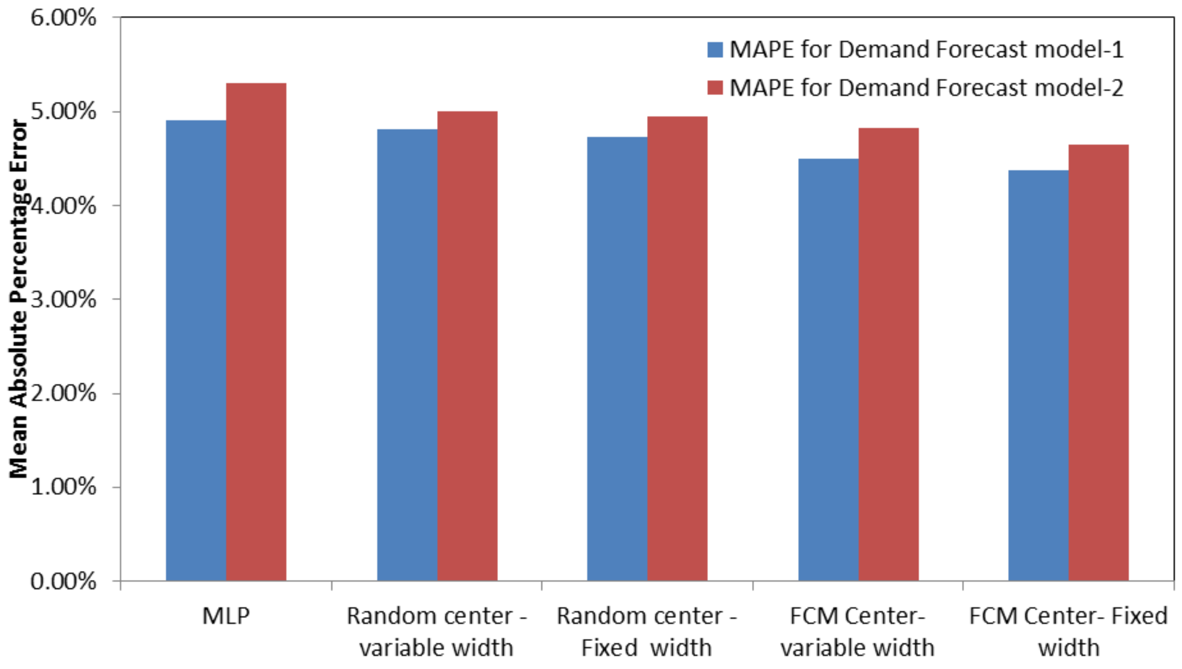


Fig: 4.23 Comparison of forecasting accuracy for MLP and different RBF network for model 1 & 2.

From Table 4.27 and Fig 4.23, we can observe that the performance of both the MLP and RBF networks is comparable, as both are robust and accurate in estimating the demand forecast. MLP requires less number of hidden units when compared to RBF networks, for the same level of performance. The generalization capability of FCM center selection RBF algorithm was found to be better than the other methods of RBF and MLP, as the MAPE was lowest in this case. RBF neural networks with FCM center selection have been effective in demand forecast with the MAPE consistently low at 4.38% and 4.65%, respectively for model 1 and model 2 for test data.

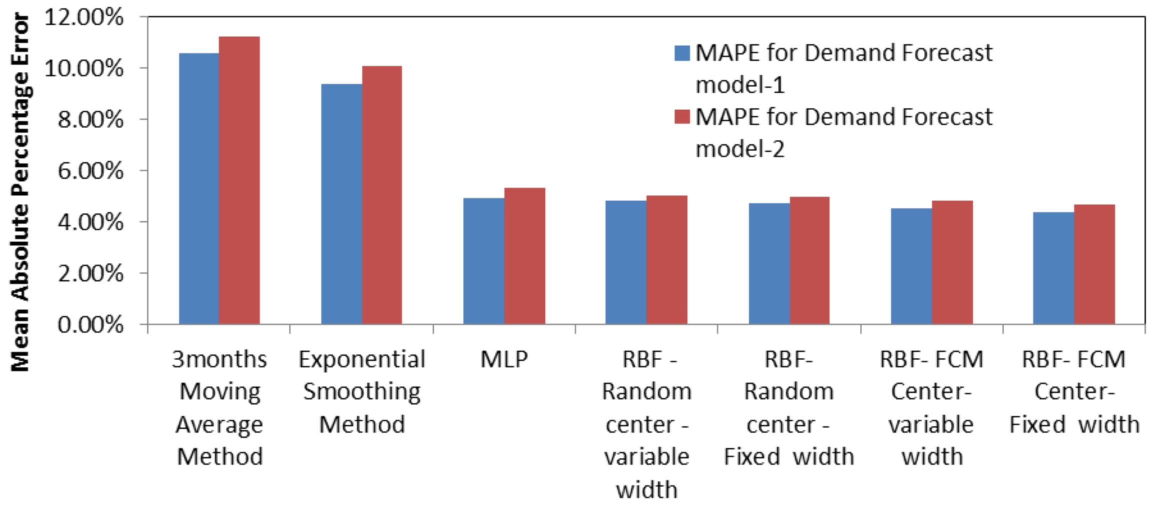
4.4 COMPARISON OF PREDICTION ACUURACY BY NEURAL NETWORK MODELS AND TRADITIONAL METHODS OF DEMAND FORECAST

The traditional demand forecast methods of 3 months moving average method and exponential smoothing methods were applied for the past historic sales data for the two types of valves, here referred as model 1 and model 2. The resulting data have been tabulated and compared with the forecasting accuracy of different ANN networks**.** Table 4.28 presents the MAPE between the demand forecast and actual sales for different forecasting methods for different forecast models. Fig 4.24 helps to pictorially visualize and compare the prediction accuracy of different traditional methods with that of neural network approaches. The traditional methods of 3 monthly moving average and exponential smoothening could predict only upto the accuracy of 10.56% and 9.39% respectively Numeral network approach has registered the forecast accuraly upto 4.38% MAPE.

It can be seen from the Table 4.28 and Fig 4.24 that the forecast accuracy of the ANN models is far superior to traditional methods. This can be attributed to superior functional approximation capability of machine learning technique and their capability to analyse the non linear and noisy data.

Table 4.28 Comparison of Prediction Accuracy By Neural Network Models And Traditional Methods Of Demand Forecast.

Demand Forecast method	MAPE for Demand Forecast model-1	MAPE for Demand Forecast model-2
3months Moving Average Method	10.56%	11.22%
Exponential Smoothing Method	9.39%	10.1%
MLP	4.91%	5.3%
Random center - variable width	4.81%	5.0%
Random center - Fixed width	4.73%	4.95%
FCM Center- variable width	4.50%	4.82%
FCM Center- Fixed width	4.38%	4.65%



Different Demand forecast ANN algorithm

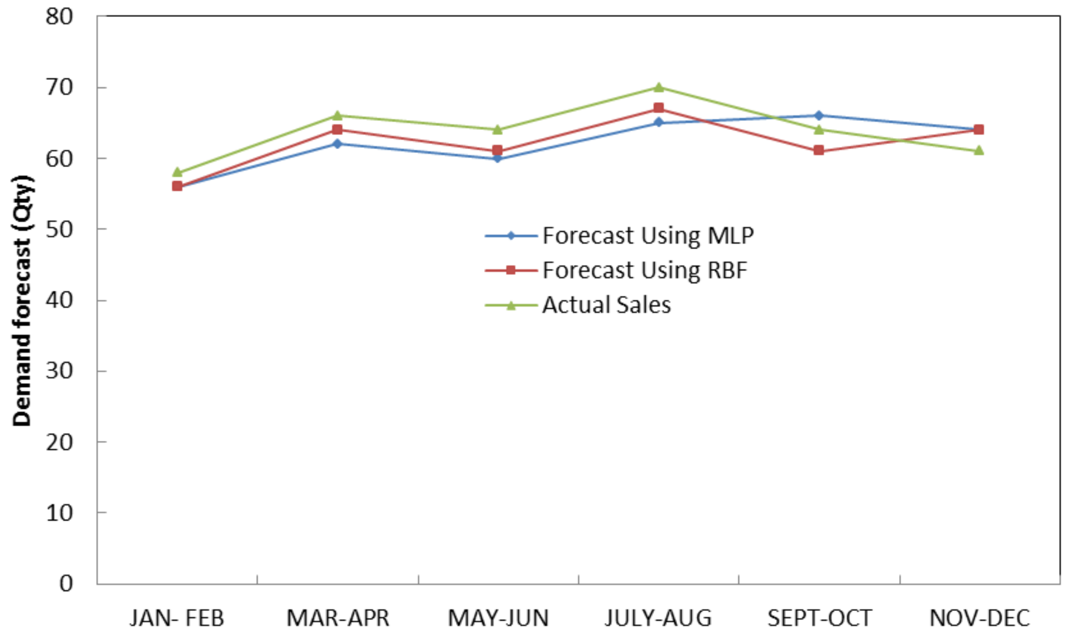
Fig 4.24 Comparison of Prediction Accuracy by Neural Network Models and Traditional Methods Of Demand Forecast.

ANN MODEL VALIDATION

The bimonthly sales data for the year 2014 for the two types of valves under study is compared with demand forecast using ANN with MLP and RBF architecture. The results are shown in Table 4.29 and Fig 4.25 for 10’’x 150 GTV101 Series Valves and Table 4.30 and Fig 4.26 for 6’’x 600 class GTV 102 Series Valves. For every two months duration, the sales demand values are predicted using MLP and RBF neural network models. These values are recorded against the actual sales in the Table 4.29 for 10’’x 150 GTV101 Series, and in Table 4.30 for 6’’x 600 class GTV 102 Series Valves. Then percentage absolute error is calculated for both MLP and RBF prediction with reference to actual sales figures. Fig 4.25 and Fig 4.26 show the graphical representation with bimonthly time intervals on X-axis and corresponding demand and sales figures on Y-axis for 10’’x 150 GTV101 Series valves and for 6’’x 600 class GTV 102 Series Valves respectively. Both Fig 4.26 and Fig 4.26 compare the 3 plots- demand figures as predicted by MLP architecture, demand figured as predicted by FBF architecture and actual sales.

Table 4.29 Demand Forecast for 10’’x 150 GTV101 Series Valves ANN Model Validation (For Year 2014).

Demand Forecast (Qty)					
	Using MLP	Using RBF	Actual sales(Qty)	MLP Absolute Error %	RBF Absolute Error %
JAN- FEB	56	56	58	3.45	3.45
MAR-APR	62	64	66	6.06	3.03
MAY-JUN	60	61	64	6.25	4.69
JULY-AUG	65	67	70	7.14	4.29
SEPT-OCT	66	61	64	3.13	4.69
NOV-DEC	64	64	61	4.92	4.92
			Mean Absolute Error %	5.16	4.18

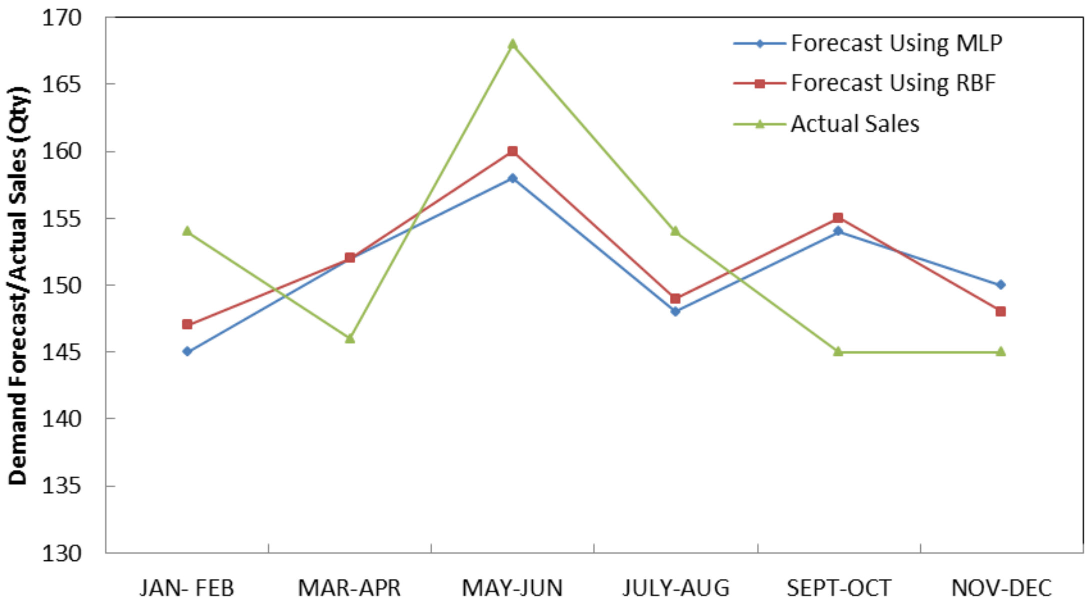


Bimonthly time period

Fig 4.25 Demand Forecast For 10''x 150 GTV 101 Series Valves ANN Model Validation (For Year 2014).

Table 4.30 Demand Forecast For 6''x 600 class GTV 102 Series Valves ANN Model Validation (For Year 2014).

Time Peroiod	Demand forecast (Qty) Using MLP	Demand forecast (Qty) Using RBF	Actual Sales (Qty)	MLP Abs. Error %	RBF Abs. Error %
Jan- Feb	145	147	154	5.84	4.55
Mar-Apr	152	152	146	4.11	4.11
May-Jun	158	160	168	5.95	4.76
Jul-Aug	148	149	154	3.90	3.25
Sept-Oct	154	155	145	6.21	6.90
Nov-Dec	150	148	145	3.45	2.07
			Mean Absolute Error %	**4.91**	**4.27**



Bimonthly time period

Fig 4.26 Demand Forecast For 6’’x 600 class GTV 102 Series Valves :ANN Model Validation (For Year 2014).

It is clear from the Table 4.29 and Fig 4.25 that the ANN model 1 for demand forecast of 10’’x 150 GTV 101 Series Valves has given a prediction accuracy, in terms of MAPE, of 5.16% and 4.17% respectively with MLP and RBFNN architecture. ANN model 2 for demand forecast of 6’’x 600 class GTV 102 Series Valves has given corresponding MAPE of 4.91 % & 4.27%, as illustrated in Table 4.30 and Fig 4.26. These results validate our model and show that they are robust enough to be used in real world industrial application. It has also proved that RBFNN models are consistently outperforming MLP models in terms of better prediction accuracy**.**

Chapter 5.

RESULTS AND DISCUSSIONS –PART II

MULTI ITEM MULTI PERIOD LOT SIZING - ACO MODELLING

5.1 INTRODUCTION

Different ANN models for demand forecast have been discussed in chapter 4. Accurate demand forecast is an important input to the dynamic lot sizing optimization problem. In this chapter, the multi objective optimization ACO models developed have been presented. The total cost of the inventory control system comprises of total holding cost, total ordering cost, and total purchasing cost. Objective is to find out the optimum values of order quantities for multi-item multi-period in order to reduce total cost of the inventory control system. A mathematical programming model is developed for inventory control systems, in which price break discount is also considered. Demand rates which are determined from ANN models discussed in previous chapter are used. Budget and warehouse area constraints are considered in order to make the model more realistic.

The developed mathematical model is a multi objective optimization problem which is NP-hard (non-deterministic polynomial-time hard**)**. Exact methods cannot solve the large-size problems of the multi-objective mixed-integer linear programming. So meta-heuristic, novel approach through Ant Colony Optimisation is proposed to solve the large-size problems. ACO algorithm application results for small size problems are matched with the results acquired from exact methods like LINGO optimization software, the comparison of which helps in verifying and validating the performance of the proposed ACO algorithm.

Further meta-heuristic, namely Genetic algorithm solution is also developed to solve the problem as well, because no benchmarks can be found in the literature to evaluate the performance of the proposed novel ACO method. Both the methods are compared on the basis of evaluation parameters like best objective value function, minimum computational time and less spread between best and worst solution in pareto front (Ali Roozbeh Nia et al. 2014).

5.2 MATHEMATICAL FORMULATION

The company maintains the stock of several items in order to satisfy its customers' known demand rates. The customers' known demand rates may change in different periods within a finite planning horizon having N periods as estimated from the demand forecast models detailed in last chapter. The initial inventory of all items is the reserve stock. Only one order is placed for a particular item in a given period. Lower limit and upper limit of order quantity for each item is also specified. Price discount breakpoints are also defined, so that, if an item is ordered in price break 1 then no discount will be offered. But if an item is ordered in price break 2, then 5% discount will be offered and if an item is ordered in price break 3, then additional 5% discount will be offered and so on. Moreover, the storage space to hold the stock for each period is constrained and the available budget for each period is also constrained. The objective is to find out the optimum values of order quantities of all the items for all the periods such that the total cost of the inventory control system is minimized and the constraints are satisfied. The concept explained above can be applied to many real-world inventory control systems and hence detailed research is carried out in this direction. The mathematical formulation of the problem is presented here (Seyed, M.M et al. 2013).

The inventory of the product u is depicted in three typical intervals. The beginning inventory of product u in period v + 1 is equal to the sum of initial inventory and the purchased quantity minus its demand, all in interval v. In other words

$I_{u,v+1} = I_{u,v} + O_{u,v} - R_{u,v}$ (1) (Seyed, M.M et al. 2013)

Where $I_{u,v}$ is the inventory of item u in the beginning of period v , $O_{u,v}$ is the quantity of item u arrived in the interval v and R_{uv} is the demand of the item u for the period v.

The total cost of the inventory control problem comprises of total purchasing cost, total holding cost, and total ordering cost, i.e., IC = OC + HC + PC (2) (Seyed, M.M et al. 2013)

The total ordering cost OC is obtained by

$$\mathrm{OC} = \sum_{u=1}^{c} \sum_{v=1}^{p} \mathrm{G_u Q_{u,v}} \quad (3)$$

Where Q_{uv} is a binary variable, with the value equal to one if the component 'u' is ordered in period v, value equal to zero otherwise.

Holding cost is the product of per unit holding cost and average inventory during that period.

Holding cost = unit holding cost x Average inventory (4) (Brahimi, N. et al. 2006)

Average inventory = ½ x (initial inventory during the start of period + remaining inventory at the period end). (5)

Using equation (1) and (5)

$$\text{Holding cost} = \textstyle\sum_{u=1}^{c} \sum_{v=1}^{p} H_u (\mathrm{I_{u,v}} + \mathrm{O_{u,v}} - \mathrm{R_{u,v}}/2) \quad (6)$$

$B_{u,v,w}$ is a binary decision variable which is equal to one if component u is purchased at price breakpoint w in period v, and zero otherwise. As a result, the total purchasing cost will be

$$\sum_{u=1}^{c} \sum_{v=1}^{p} \sum_{w=1}^{d} \mathrm{C_{u,w}\ O_{u,v}\ B_{u,v,w}} \quad (7)$$

As a result, the mathematical model of the inventory control system is as follows.

Minimize the total cost or the objective function

$$Z = \sum_{u=1}^{c}\sum_{v=1}^{p} G_u\ Q_{u,v} + \sum_{u=1}^{c}\sum_{v=1}^{p} H_u\ (I_{u,v} + O_{u,v} - R_{u,v}/2) + \sum_{u=1}^{c}\sum_{v=1}^{p}\sum_{w=1}^{d} C_{u,w}\ O_{u,v}\ B_{u,v,w} \qquad (8)$$

(Source: Seyed, M.M et al. 2013).

The constraints are as follows for u=1, 2,…,c. v=1, 2,...,p and w=1, 2,...,d.

1. $Q_{u,v}$ = 0 or 1 (Boolean value), $B_{u,v,w}$ = 0 or 1 (Boolean value)

2. $\sum_{w=1}^{d} B_{u,v,w}=1$

The above constraint ensures that the quantity should be bought at only one price break.

3. $I_{u,v+1} = I_{u,v} + O_{u,v} - R_{u,v}$

The informal meaning of the above constraint is that inventory brought forward to next period = inventory brought forward to this period + quantity ordered in this period - demand (or consumption) in this period.

4. $I_{u,v} \geq S_u$

The above constraint ensures that the inventory brought forward should be greater than or equal to the reserve stock.

5. $\sum_{u=1}^{c} C_u\ O_{uv} \leq A_v$ for v=1, 2,...p.

The above equation ensures that budget constraint should be satisfied for each period.

6. $\sum_{u=1}^{c} m_u\ (I_{uv} + O_{uv}) <= M$ for v=1, 2,...p.

The above equation ensures that warehouse area constraint should be satisfied for each period.

In this type of problem of mixed-integer linear programming model the computation time will increase enormously with the increase in the size of problem in polynomial order. Thus, for solving large-size problems, a population based meta heuristic method like ACO is proposed.

5.3 ACO ALGORITHM MODELLING

One of the most successful methods of swarm intelligence is ant colony optimization. Marco Dorigo invented ant colony optimization during his research work. It was created as a probabilistic method that could be utilized to solve optimization problems (Chen-Yang Cheng et al.2015). Ant colony optimization algorithm mimics the principle of biological or real ant colony system. The flow chart of meta-heuristic ant colony optimization to solve inventory control problem is shown in figure 3.12 and detailed description is given in section 3.10 to narrate how the ACO model has been developed for the inventory control problem.

5.4 COMPUTATIONAL RESULTS

This section briefly explains the computational procedure and presents the results of the numerical studies to evaluate and validate the developed ACO model.

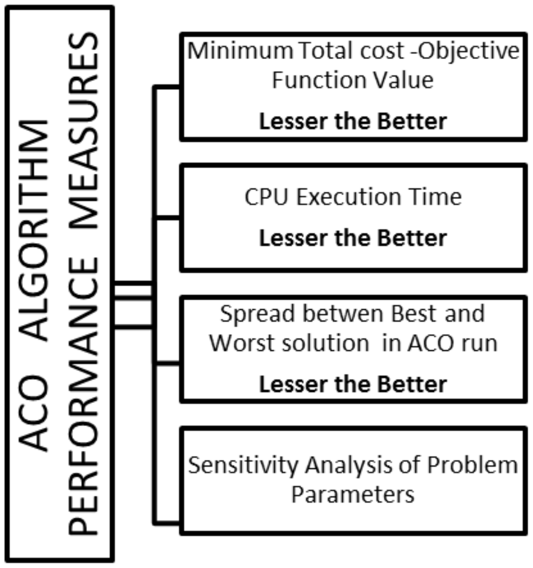

ACO ALGORITHM PERFORMANCE MEASURES
Minimum Total cost -Objective Function Value
Lesser the Better
CPU Execution Time
Lesser the Better
Spread betwen Best and Worst solution in ACO run
Lesser the Better
Sensitivity Analysis of Problem Parameters

Fig 5.1 ACO model performance measure.

Fig 5.1 shows the different performance measures which will be used to test the capability of the ACO model. The performance of developed model can be evaluated based on 3 important parameters namely, minimum total cost, minimum CPU time of execution, minimum spread between best and worst solution. In addition, best sensitivity to different problem parameter is another platform to evaluate the effectiveness of the model. [Ali Diabat et al. 2016, Ba-Yi Cheng et al. 2010, Behnam Vahdani et al. 2017]

In order to demonstrate the application of the developed ACO, and investigate their performance in suggesting the most economical periodical lot sizing taking into consideration dynamic demand, ordering, holding and purchase cost, 8 numerical examples are considered with 8 different data sets. The data input is from the real industry scenario under study. Data for six most important valve configuration are studied in depth and tabulated as in Table 5.1 to 5.3. For ease of identification, the data sets are numbered from 1to 8. Each data set can be conveniently represented as data set no. (c-p-w) where c is the number of items for procurement, p is the period and w is the number of price breaks for discount. The tables contain data on bimonthly projected demand of items under study. Information on holding cost , ordering cost, purchase cost, reserve stock , budget and warehouse constraints are also listed in the above tables for different data sets.

ACO was run 4 times for each problem configuration. For each ACO run, the order quantities of every item for each period are tabulated. Minimum total cost or the value of objective function which is an important performance measure is listed. The CPU execution time is also noted for each ACO run. The spread between best and worst solution is also marked for each ACO run. The lesser the spread, better is the algorithm. All the test problems are solved on a lap top with Intel core i3-2100 processor having 3.10 GHz CPU and 4 Gig RAM.

5.4.1 Setting ACO program Parameter

The number of cycles which is used for ACO run plays an important role in determining the quality of optimization result and also the time taken to arrive at the result. Initially increasing the number of cycles may improve the quality of result. After reaching a certain threshold value, additional number of cycles would only increase the CPU time taken for the cycle execution without adding any value for improvement of the quality of the solution.

To arrive at the optimum value of number of cycles, which is an important ACO algorithm parameter, data set no. 4 (3-3-3) is run several times changing the number of cycles each time and recording the objective value function and CPU processing time (Ba-Yi Cheng et al. 2010).

Each row of Table 5.4 records number of cycles adopted and corresponding objective function value obtained and CPU execution time. Same data is presented in the graphical form in Fig 5.2 marking number of cycles on X axis and objective function value on Y axis. Fig. 5.3 contains the plot of number of cycles vs the CPU execution time. Results are tabulated in Table 5.4.

When the number of cycles used is 500, the objective function value is Rs 8088430/- and CPU execution time is 41sec. The number of cycles adopted is increased by 100. The objective function value improved to Rs 8082510/- and CPU execution time is 54. When the number of cycles is increased to 700, there is a marginal improvement in the objective function value to Rs8082435/-. The CPU execution time is drastically increased to 62 sec.

From the Table 5.2 and Fig. 5.2 & Fig.5.3, it is clear that objective value reaches optimum value at 600 cycles. After that, increasing the no. of cycles will only increase the CPU time without improving the objective function value.

Table 5.1 Input data for Lot Sizing Problem: Data set no. 1,2,3,4&5.

Data set no. 1 c=1, p=3, w=2	
C	1
p	3
w	2
R11	40
R12	70
R13	50
G1	25000
H1	1020
C11	6025
C12	5723.75
S1	20
A1	5000000
A2	5000000
A3	5000000
m1	0.25
M	300

Data set no.2 c=1, p=3, w=3	
C	1
p	3
w	3
R11	40
R12	70
R13	50
G1	25000
H1	1020
C11	6025
C12	5723.75
C13	5422.5
S1	20
A1	5000000
A2	5000000
A3	5000000
m1	0.25
M	300

Data set no. 3 c=2, p=3, w=2	
c	2
p	3
w	2
R11	40
R12	70
R13	50
R21	130
R22	110
R23	140
G1	25000
G2	20000
H1	1020
H2	500
C11	6025
C12	5723.75
C21	5100
C22	4845
S1	20
S2	40
A1	5000000
A2	5000000
A3	5000000
m1	0.25
m2	0.2
M	300

Data set no. 4	
c	2
p	3
w	3
R11	40
R12	70
R13	50
R21	130
R22	110
R23	140
G1	25000
G2	20000
H1	1020
H2	500
C11	6025
C12	5723.75
C13	5422.5
C21	5100
C22	4845
C23	4590
S1	20
S2	40
A1	5000000
A2	5000000
A3	5000000
m1	0.25
m2	0.2
M	300

Data set no. 5 c=3, p=3, w=3	
c	3
p	3
w	3
R11	40
R12	70
R13	50
R21	130
R22	110
R23	140
R31	300
R32	260
R33	280
G1	25000
G2	20000
G3	9000
H1	1020
H2	500
H3	425
C11	6025
C12	5723.75
C13	5422.5
C21	5100
C22	4845
C23	4590
C31	4575
C32	4346.25
C33	4117.5
S1	20
S2	40
S3	85
A1	5000000
A2	5000000
A3	5000000
m1	0.25
m2	0.2
m3	0.15
M	300

(c= no. of items, p= no. of periods, w= no. of price breaks).

Table 5.2 Input data for Lot Sizing problem: Data set no. 6 & 7.

Data set no.6 (4-3-3) c=4, p=3, w=3			
c	**4**	C31	4575
p	**3**	C32	4346.25
w	**3**	C33	4117.5
R11	40	C41	3550
R12	70	C42	3372.5
R13	50	C43	3195
R21	130	S1	20
R22	110	S2	40
R23	140	S3	85
R31	300	S4	30
R32	260	A1	5000000
R33	280	A2	5000000
R41	80	A3	5000000
R42	92	m1	0.25
R43	98	m2	0.2
G1	25000	m3	0.15
G2	20000	m4	0.1
G3	9000	M	300
G4	5000		
H1	1020		
H2	500		
H3	425		
H4	300		
C11	6025		
C12	5723.75		
C13	5422.5		
C21	5100		
C22	4845		
C23	4590		

Data set no.7 (5-3-3) c=5, p=3, w=3			
c	**5**	C12	5422.5
p	**3**	C13	5121.25
w	**3**	C21	5100
R11	40	C22	4590
R12	70	C23	4335
R13	50	C31	4575
R21	130	C32	4346.25
R22	110	C33	4117.5
R23	140	C41	3550
R31	300	C42	3195
R32	260	C43	3017
R33	280	C51	4500
R41	80	C52	4275
R42	92	C53	4050
R43	98	S1	20
R51	600	S2	40
R52	665	S3	85
R53	620	S4	30
G1	25000	S5	190
G2	20000	A1	10000000
G3	9000	A2	10000000
G4	5000	A3	10000000
G5	7000	m1	0.25
H1	600	m2	0.2
H2	300	m3	0.15
H3	425	m4	0.1
H4	200	m5	0.1
H5	150	M	600
C11	6025		

(c= no. of items, p= no. of periods, w= no. of price breaks).

Table 5.3 Input data for Lot Sizing problem : data set no. 8.

Data set no.8 (6-3-3) c=6, p=3, w=3					
c	6	H6	300	m6	0.15
p	3	C11	6025	M	600
w	3	C12	5422.5		
R11	40	C13	5121.25		
R12	70	C21	5100		
R13	50	C22	4590		
R21	130	C23	4335		
R22	110	C32	4346.25		
R23	140	C33	4117.5		
R31	300	C41	3550		
R32	260	C42	3195		
R33	280	C43	3017		
R41	80	C51	4500		
R42	92	C52	4275		
R43	98	C53	4050		
R51	600	C61	7000		
R52	665	C62	6650		
R53	620	C63	6300		
R61	425	S1	20		
R62	440	S2	40		
R63	460	S3	85		
G1	25000	S4	30		
G2	20000	S5	190		
G3	9000	S6	130		
G4	5000	A1	10000000		
G5	7000	A2	10000000		
G6	7000	A3	10000000		
H1	600	m1	0.25		
H2	300	m2	0.2		
H3	425	m3	0.15		
H4	200	m4	0.1		
H5	150	m5	0.1		

(c= no. of items, p= no. of periods, w= no. of price breaks).

.

Table 5.4 Variation of Objective function value and Time of execution with No. of cycles.

No. of cycles	Objective function value (Rs)	Time of execution(Sec)
50	8382155	5
100	8255916	9
200	8101560	20
300	8094680	32
500	8088430	41
600	8082500	54
700	8082435	62
800	8082440	75
1000	8082440	90

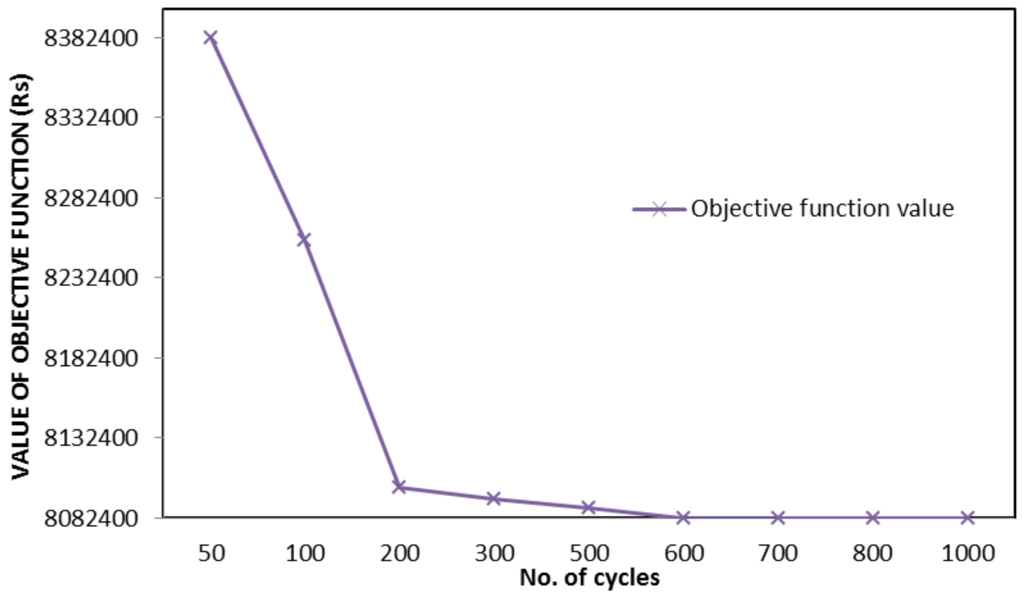


Fig 5.2 Variation of objective function value with no. of cycles- ACO model.

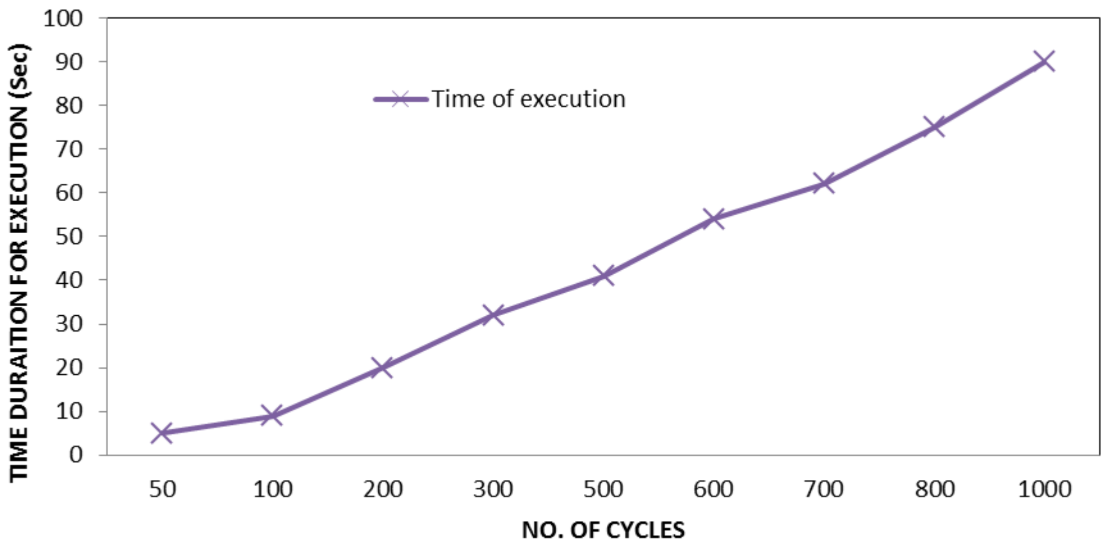


Fig 5.3 Variation of Time of execution with number of cycles- ACO model.

5.4.2 Results analysis

Table 5.5 shows the ACO run results for data set no. 1, where one product, 3 periods and 2 price breaks are considered. Table 5.6 shows the results of ACO run for data set no. 2, where 1 component, 3 periods and 3 price break point are considered for lot sizing. Table 5.7 & 5.8 tabulates the results for 2 products. The results shown in the table are the records of the order quantities for each period, best solutions obtained for total cost of inventory or objective function value in the cost minimization model and CPU execution time for each iteration and spread of between best and worst solution. ACO is run four times. But same result is obtained all the 4 times. Our ACO prescribed the same optimal solution each time, as that is obtained via exact methods using the LINDO 14 Linear Integer Programming software. Since this is a case of simple problem which involves small number of variables, the solution obtained using ACO is the exact or the most optimal one. Solving the above problem using other optimization software like GAMS or LINDO also gives the same result. This fact validates model (Hamed Soleimani et al. 2015). Having verified its ability to optimize the lot sizing, ACO can now be used to solve the complex problems having more number of decision variables. However, exact methods cannot be used for large size problem because of very large number of variables involved, the program becomes unstable or takes very long time.

Similar strategy for model validation was adopted by Fardin et al. 2015, Hamed Soleimani et al. 2015, Ata Allah Taleizadeh et al. 2013, in their research work related to application of soft computing techniques to supply chain management. Hamed Soleimani validated his Particle swarm optimisation and GA model of closed loop supply chain network by matching the results obtained by their proposed algorithm with that acquired from exact methods like LINGO using small size problem instances. Fardin used the same approach of comparing the results of exact method from GAMS software to evaluate and validate the performance of their hybrid GA, for three echelon supply chain problem.

When only one discount is considered the order quantity lot was 51, 59 & 50 for the 3 periods, as in Table 5.5 and the minimum total cost was Rs 1159882/- . When 2 discounts or 3 price breaks are considered as in data set no. 2, the most economical lot was found to be 40, 120, 0 as can be seen from the table 5.6.

Table 5.5 ACO run results for Data set no. 1: c=1, p=3, w=2.

Run no.	Item no.	Order Qty. (No) Period 1	Order Qty. (No) Period 2	Order Qty(No) Period3	Best Sol. Total Cost (Rs)	Spread bet. Best and worst solution (Rs)	CPU time (Sec)
1	1	51	59	50	1159882/-	144660/-	12
2	1	51	59	50	1159882/-	144660/-	12
3	1	51	59	50	1159882/-	144660/-	12
4	1	51	59	50	1159882/-	144660/-	12

Table 5.6 ACO run results for Data set no. 2: c=1, p=3, w=3.

Run no.	Item no.	Order Qty.(No) Period 1	Order Qty(No) Period 2	Order Qty(No) Period3	Best Sol Total Cost(Rs)	Spread bet. Best and worst solution	CPU time (Sec)
1	1	40	120	0	1135500/-	121445/-	12
2	1	40	120	0	1135500/-	121445/-	12
3	1	40	120	0	1135500/-	121445/-	12
4	1	40	120	0	1135500/-	121445/-	12

Table 5.7 ACO run results for data set no. 3: c=2, p=3, w=2.

Run no.	Item no.	Order Qty.(No) Period 1	Order Qty(No) Period 2	Order Qty(No) Period3	Best Sol Total Cost(Rs)	Spread bet. Best and worst solution	CPU time (Sec)
1	1	51	59	50	3215982/-	336700/-	20
	2	130	110	140			
2	1	51	59	50	3215982/-	336700/-	20
	2	130	110	140			
3	1	51	59	50	3215982/-	336700/-	20
	2	130	110	140			
4	1	51	59	50	3215982	336700	20
	2	130	110	140			

Table 5.8 ACO run results for data set no. 4: c=2, p=3, w=3.

Run no.	Item no.	Order Qty. (No). Period 1	Order Qty. (No). Period 2	Order Qty. (No). Period3	Best Sol Total Cost(Rs)	Spread bet. Best and worst solution	CPU time (Sec)
1	1	40	120	0	3494700/-	316015/-	22
	2	130	110	140			
2	1	40	120	0	3494700/-	316015/-	22
	2	130	110	140			
3	1	40	120	0	3494700/-	316015/-	22
	2	130	110	140			
4	1	40	120	0	3494700/-	316015/-	22
	2	130	110	140			

Table 5.9 ACO run results for data set no. 5: c=3, p=3, w=3.

Run no.	Item no.	Order Qty. (No). Period 1	Order Qty.(No) Period 2	Order Qty. (No) Period3	Best Sol. Total Cost (Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	1	42	118	131	7015841/-	493352/-	40
	2	131	120	129			
	3	361	208	271			
2	1	40	120	0	7010875/-	497825/-	40
	2	134	108	138			
	3	372	468	00			
3	1	43	117	0	7017807/-	485675/-	40
	2	138	102	140			
	3	372	468	0			
4	1	41	119	0	7021825/-	491372/-	40
	2	130	110	140			
	3	377	426	377			
Average Objective Fn value & Avg. spread					7016587/-	492056/-	40

Table 5.10 ACO run results for Data set no. 6: c=4, p=3, w=3.

Run no.	Item no.	Order Qty.(No) Period 1	Order Qty.(No) Period 2	Order Qty.(No) Period3	Best Sol. Total Cost(Rs)	Spread bet. Best and worst solution (Rs)	CPU time (Sec)
1	1	41	103	16	8070056/-	487646/-	62
	2	141	109	130			
	3	375	214	251			
	4	270	0	0			
2	1	60	53	47	8072996/-	469560/-	62
	2	139	103	138			
	3	361	414	65			
	4	270	0	0			
3	1	51	109	0	8082756/-	485410/-	62
	2	144	112	124			
	3	377	401	62			
	4	270	0	0			
4	1	54	106	0	8073832/-	489540/-	62
	2	131	116	133			
	3	380	402	58			
	4	270	0	0			
Average Objective Fn value & Avg. spread					8074910/-	483039/-	62

Table 5.11 ACO run results for data set no. 7: c=5, p=3, w=3.

Run no.	Item no.	Order Qty.(No) Period 1	Order Qty(No) Period 2	Order Qty(No) Period3	Best Sol Total Cost(Rs)	Spread bet. Best and worst solution (Rs)	CPU time (Sec)
1	1	160	0	0	15718292/-	755940/-	75
	2	380	0	0			
	3	361	244	235			
	4	270	0	0			
	5	756	654	475			
2	1	51	109	0	16447257/-	720387/-	75
	2	159	211	10			
	3	370	233	237			
	4	270	0	0			
	5	671	756	458			
3	1	157	3	0	15726613/-	745871/-	75
	2	380	0	0			
	3	365	211	264			
	4	110	144	16			
	5	815	504	566			
4	1	53	107	0	15702395/-	766744/-	75
	2	153	227	0			
	3	307	253	280			
	4	270	0	0			
	5	663	711	511			
Average Objective Fn value & Avg. spread					15899639/-	747235/-	75

Table 5.12 ACO run results for data set no. 8: c=6, p=3, w=3.

Run no.	Item no.	Order Qty.(No) Period 1	Order Qty.(No) Period 2	Order Qty. (No) Period3	Best Sol. Total Cost (Rs)	Spread bet. Best and worst solution (Rs)	CPU time (Sec)
1	1	46	114	0	25321202/-	645500/-	80
	2	153	210	17			
	3	377	411	52			
	4	260	6	4			
	5	628	721	536			
	6	480	413	432			
2	1	72	55	33	25331901/-	622969/-	80
	2	160	220	11			
	3	377	380	83			
	4	268	2	0			
	5	635	648	602			
	6	444	425	456			
3	1	52	108	0	25324673/-	651288/-	80
	2	203	161	16			
	3	364	262	214			
	4	104	146	20			
	5	616	680	589			
	6	429	440	456			
4	1	42	118	0	25301616/-	680659/-	80
	2	201	179	0			
	3	366	244	230			
	4	270	0	0			
	5	637	658	590			
	6	426	456	443			
Average Objective Fn value & Avg. spread					25324848/-	650104/-	80

Table 5.13 ACO run performance parameter for 8 different programs

Problem no.	Problem Description	Best Sol Total Cost (Rs)	Spread bet. best and worst solution(Rs)	CPU time (Sec)
1	c=1, p=3, w=2	1159882/-	144660/-	12
2	c=1, p=3, w=3	1135500/-	121445/-	12
3	c=2, p=3, w=2	3215982/-	336700/-	20
4	c=2, p=3, w=3	3494700/-	316015/-	22
5	c=3, p=3, w=3	7016587/-	492056/-	40
6	c=4, p=3, w=3	8068146/-	509588/-	62
7	c=5, p=3, w=3	15899639/-	747235/	75
8	c=6, p=3, w=3	25324848/-	650104/-	80

The minimum total cost also came down to Rs1135500/- , a saving of around 5%, due to savings from one less order and higher discount above the price break quantity of 100 units, in spite of higher carrying cost expenses.

For lot sizing decisions for 3 components and above, the number of decision variables increases and we can find different solutions in different ACO runs. This is evident from table 5.9 to 5.12. ACO run results for data set no. 5 (3-3-3), data set no 6(4-3-3), data set no. 7(5-3-3) and data set no. 8(6-3-3) are tabulated in Table 5.9, Table 5.10, Table 5.11 & Table 5.12 respectively. Four data sets are considered with same number of periods and price breaks but different number of items. Above tables show the ACO run results in terms of order quantities and optimal total cost and CPU processing time.

Table 5.11 refers to the result obtained for data set which reflects a scenario of inventory control where four components are considered. The ACO algorithm optimizes the quantities of procurement of these components when three periods and three price breaks are taken into account. Results from four different runs of ACO algorithm are tabulated. First run specifies that the item no.1 has to be procured in the lot sizes of 41, 103 and 16 respectively during the period 1, 2 and 3. Item 2 should be bought in the lot sizes of 141, 109 and 130. Recommended lot size for item 3 and 4 are 375, 214, 251 and 270,0, 0, respectively in the period 1, 2 and 3. The total cost of procurement amounts to Rs. 8070056/-. This lot size has been suggested by the ACO algorithm considering the carrying cost, ordering cost and purchase cost of all the four components, thereby optimizing the total cost of inventory. The limiting constraints of total annual budget, warehouse area and reserve stock have also been taken into account.

Each run of ACO consists of 600 cycles. Each cycle suggests the order quantities and total cost. Among these 600 values, minimum total cost and corresponding order quantities have been recorded. The spread between the best and worst solution is also recorded which gives an indication of stability of algorithm and robustness of the solution from this population based meta heuristic algorithm.

The average optimal cost for the data set no. 5 (3-3-3), data set no. 6 (4-3-3), data set no. 7 (5-3-3) and data set no. 8 (6-3-3) are Rs 7016587/-, Rs 8068146/-, Rs15899639/- and Rs 25324848/-, respectively as seen from tables to 5.9 to 5.12. The CPU time also increases from 12 sec for 3 items to 80 sec for 6 items. Table 5.13 lists the ACO run performance parameter for 8 different data sets.

5.5 SENSITIVITY ANALYSIS

Sensitivity analysis is the key to performance appraisal of the mathematical models and their solution (Devendra Choudhary et al. 2011). Goh Sue et al. (2012) has used the sensitivity analysis to evaluate the performance of Particle Swarm Optimisation based heuristics in determining the optimal solution for vendor managed inventory control problems in multi echelon supply chain. Devendra Choudhary et al. (2011) used the sensitivity of problem parameters to prove the effectiveness of their novel Linear Integer based program in determining the dynamic lot size for single item multi period procurement.

In this section, the effect of variation in problem parameters is investigated on multi-period lot-sizing decision from our algorithm. The ordering cost, holding cost, purchase cost in terms of discounts and price breaks are the major input parameters for the proposed algorithm.

5.5.1 Sensitivity to ordering cost

Data set no. 1 (c=1, p=3, w=2) is used to study the effect of ordering cost on the procurement lot size and total cost by keeping all the data same as in Table 5.6, except the ordering cost Gu. Table 5.14 lists the different periodic lot size by running the ACO by varying the ordering cost.

Table 5.14 Ordering cost –sensitivity analysis- ACO model.

Run no.	Ordering cost (Rs)	Order Qty. (No.) Period 1	Order Qty (No.) Period 2	Order Qty. (No.) Period3	Best Sol Total Cost (Rs)
1	25000/-	51	59	50	1159882/-
2	35000/-	51	59	50	1189882/-
3	35940/-	51	109	0	1192700/-
4	111180/-	160	0	0	1343180/-

Fig. 5.4 graphically shows the effect of varying the ordering cost on the order quantities suggested by the model through bar chart. It can be seen from Table 5.14 and Fig. 5.4, that the algorithm suggested order quantities for 3 periods when the ordering cost was less than around Rs. 35940/-. After that the lot sizing suggested to buy the items only at 2 periods reducing the number of annual order to 2. The saving in ordering cost was more than the increase in holding cost expenses.

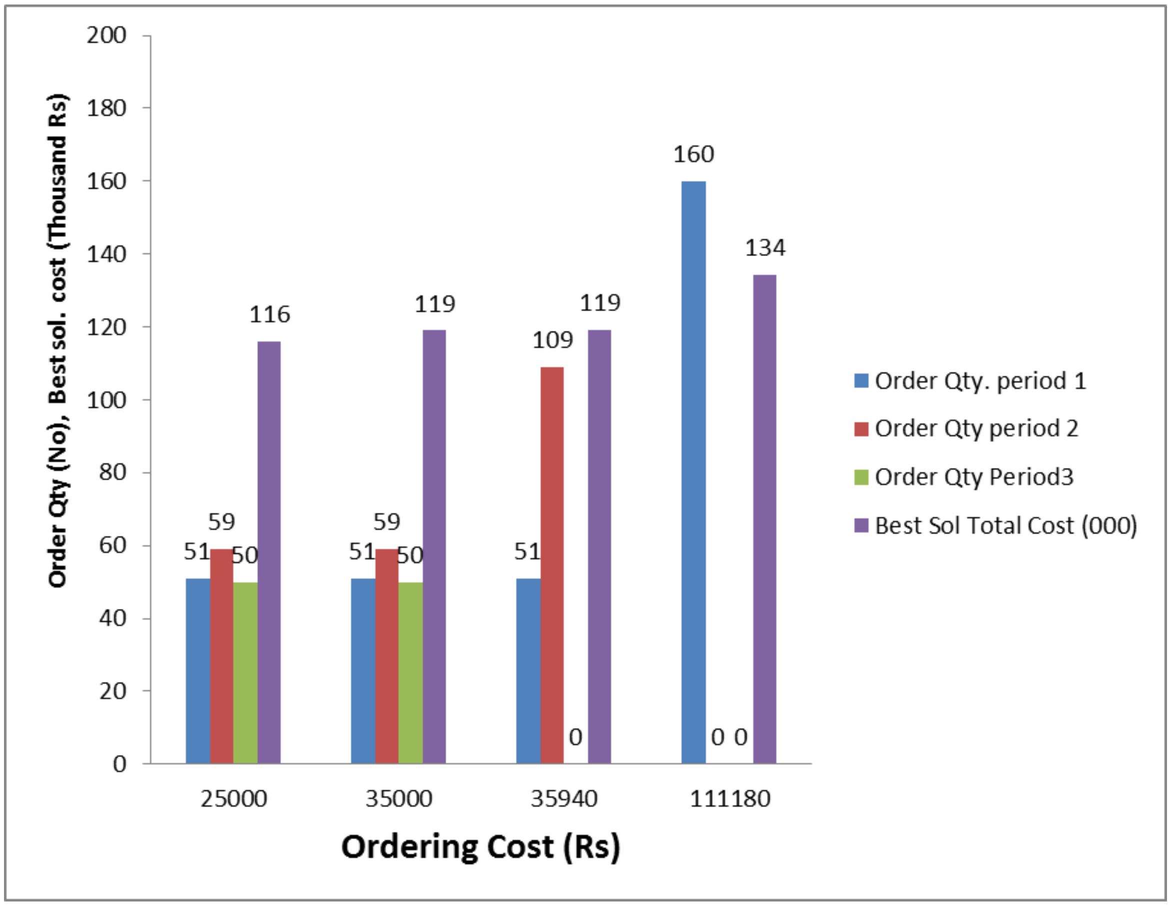


Fig 5.4 Ordering cost: Sensitivity Analysis- ACO model.

When the ordering cost is still higher at Rs 111180/- the algorithm suggests only one lot to be bought at the year beginning on the basis of ordering cost- holding cost trade off. This clearly suggests that the proposed ACO model responds well to the changes in the ordering cost

5.5.2 Holding cost sensitivity

Again data set no. 1 (c=1, p=3, w=2) is used to study the effect of holding cost on the procurement lot size and total cost by keeping all the data same as in table 5.6, except the holding cost Hu. Table 5.15 lists the different periodic lot size by running the ACO by varying the holding cost. For the present problem, at holding cost of Rs 1020/-, the order quantities are 51, 59 and 50 for the three periods, respectively. If holding cost is increased, a point will reach at around Rs 1100/-, where the GA algorithm will suggest the Just In Time(JIT) procurement of quantities 40, 70, 50 respectively for the 3 periods because at this cost, there can not be any trade off between holding and ordering cost.

Fig 5.5 shows graphically the effect of holding cost variation. The bar chart in Fig.5.5 has the order quantities marked on Y-axis. X-axis shows the holding cost. The order quantities for the 3 periods as suggested by the model at different ordering cost are displayed. Bar chart clearly shows at holding cost valve of 800/-, period 3 ordering quantity is zero. At holding cost value of 225/- both period 3 and period 2 ordering quanity value is zero.

Table 5.15 Holding cost –sensitivity analysis- ACO model.

Run no.	Ordering Cost (Rs)	Order Qty. (No.) Period 1	Order Qty (No.) Period 2	Order Qty. (No.) Period3	Best Sol Total Cost (Rs)
1	1020/-	51	59	50	1159882/-
2	1100/-	40	70	50	1171912/-
3	800/-	51	109	0	1126600/-
4	225/-	160	0	0	1010550/-

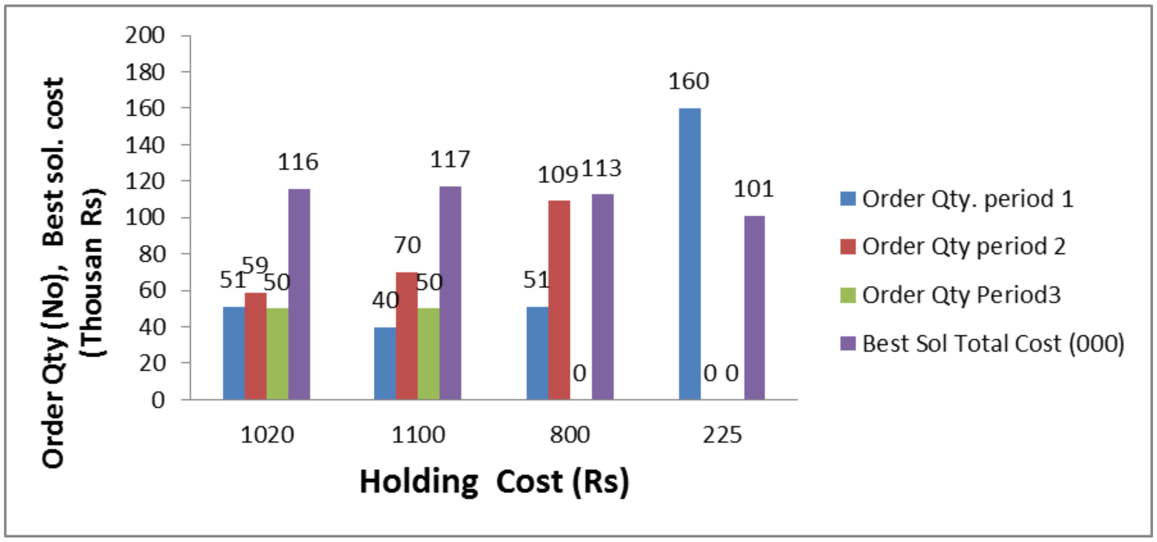


Fig 5.5 Holding Cost –sensitivity analysis- ACO model.

It will be interesting to note that as the holding cost is reduced to around Rs 800/-, first trade off is reached between ordering and holding cost. So, it becomes more advantageous to reduce one order in spite of higher carrying expenses due to larger inventory. So, we get order quantities 51, 109, 0. When the holding cost is further reduced, at around Rs 220/-, it becomes less expensive to buy all the annual demand of 160 at the beginning of the year, due to saving in ordering cost. This clearly suggests that ACO model responds well to the change in holding cost.

5.5.3. Sensitivity to discounts and price breaks

Referring to Table 5.6, problem no. 1 and 2 are same except that in problem no. 2, there is an additional price break that is specified at quantity 100 which will become eligible for discount of 10%. Comparing ACO solution to problem 1 and 2, (table 5.5 & 5.6), it is clear that order quantity has been increased to full requirement of 120 units in the second period, as there is a cost reduction due to higher discounts of 10% at the price break quantity 100 units.

Additional sensitivity exercises are carried out as per the table 5.16 taking the data from problem no.2, but varying the discounts and price break quantities.

Table 5.16 Price discounts and price break sensitivity- ACO model.

No.	Discount/ price break 1	Discount/ price break 2	Order Qty. (No) Period 1	Order Qty. (No) Period 2	Order Qty. (No) Period3	Best Sol Total Cost (Rs)
1	5% at & above 50	NIL	51	59	50	1159882/-
2	5% at & above 60	10% @ and above 100	40	120	0	1135500/-
3	25% at and above 150	Nil	160	0	0	1064200/-

In the scenario no.1, all the order quantities are selected that they are above 50 to see that they are eligible for the 5% discount at the first price break. In the scenario 3, there is single discount of 25% above price break quantity of 150. To avail this huge discount, the order quantity of 160 is suggested in the first period itself to ensure that the order quantity stays at above 150.

It is evident from above three sub sections that the proposed multi period multi item periodic lot sizing model responds to the variation in problem parameters, or in other words, it is sensitive to changes in different parameters which specify the problem. Results analyzed confirm that the suggested model responds well to all realistic constraints and tradeoffs in cost objectives. Optimal procurement lot-size is obtained by striking best tradeoffs among multiple cost objectives.

Smaller lot-size reduces inventory holding cost but increases purchasing cost and ordering cost due to lack of economy of scale. Larger lot-size reduces purchasing cost, and leads to higher inventory holding cost. All these are reflected in the sensitivity analysis.

5.6 SUMMARY OF RESULTS

1. The mathematical model is constructed for inventory management problem for optimizing multi item multi period lot sizing considering deterministic but variable demand. Model is made more realistic to suit the requirements of company under study by constraining budget and storage space.
2. Since the model is NP hard, Meta heuristic ACO algorithm is developed for the solution.
3. ACO program parameter is set by running the representative problem several times and optimizing the objective value function.
4. ACO model solution is evaluated based on performance parameters- minimum total cost of best solution which is the objective function value, CPU processing time, spread between best and worst solution & problem parameter sensitivity analysis.
5. ACO run carried out for each of 8 data set scenario and performance parameters along with order quantities are tabulated. Simple problem solutions with single item are verified against exact method which validates the proposed ACO model.
6. Comprehensive graph developed indicating the ACO model solution performance parameters for each of the problem scenario.
7. Sensitivity analysis carried out for the problem parameters like ordering cost, holding cost and price break and discount. Cost trade off scenarios are well verified.
8. It is evident from the computational results that suggested ACO model properly analyses the trade off in cost objectives and that it captures well all the realistic constraints in the decision making process for the multi period procurement lot sizing problem. The model is robust enough to accommodate the real life scenario in an industrial and trading set up for suggesting the optimum procurement lot sizes.

Chapter 6.

RESULTS AND DISCUSSIONS –PART III

MULTI ITEM MULTI PERIOD LOT SIZING- GA MODELLING

6.1 INTRODUCTION

A mixed binary integer mathematical programming model was developed in chapter 5 for ordering items in multi-item multi-period inventory control systems, where the demand rate is deterministic, but varying. A meta heuristic solution methodology ACO was also developed. Since there are no bench marking methods available in the ready literature to evaluate the performance of the ACO model, GA model is developed in this chapter and employed to solve the problem. As an established practice, the performance measures of both the algorithm are compared. One way Analysis of Variance (ANOVA) is employed to explore the statistically significant differences in the performance parameters (Javad Sadeghi et al. 2014, Seyed Mohsen Mousavi et al. 2014).

A mechanism based on the Taguchi optimization technique is used to calibrate the parameters of GA program thereby attempting to enhance the performance of GA.

6.2 MATHEMATICAL MODEL

Complete mathematical model of the inventory control system for periodic order lot sizing which has been developed in chapter 5 is as follows.

Minimize:

$$Z = \sum_{u=1}^{c}\sum_{v=1}^{p} G_u\, Q_{u,v} + \sum_{u=1}^{c}\sum_{v=1}^{p} H_u\, (I_{u,v} + O_{u,v} - R_{u,v}/2)$$
$$+ \sum_{u=1}^{c}\sum_{v=1}^{p}\sum_{w=1}^{d} C_{u,w}\, O_{u,v}\, B_{u,v,w} \qquad \text{(Seyed, M. M. et al. 2013)}$$

The constraints are as follows for u=1, 2,…c, v=1, 2,...p and w=1, 2,...d.

1. $Q_{u,v}$ = 0 or 1 (Boolean value), $B_{u,v,w}$ = 0 or 1 (Boolean value).

2. $\sum_{w=1}^{d} B_{u,v,w}=1$

The above constraint ensures that the quantity should be bought at only one price break.

3. $I_{u,v+1} = I_{u,v} + O_{u,v} – R_{u,v}$

The informal meaning of the above constraint is as follow. Inventory brought forward to next period = inventory brought forward to this period + quantity ordered in this period - demand (or consumed) in this period.

4. $I_{u,v} \geq S_u$

The above constraint ensures that the inventory brought forward should be greater than or equal to the reserve stock.

5. $\sum_{u=1}^{c} C_u O_{uv} \leq A_v$ for v=1, 2,...,p

The above equation ensures that budget constraint should be satisfied for each period.

6. $\sum_{u=1}^{c} m_u (I_{uv} + O_{uv}) <= M$ for v=1, 2,...,p

The above equation ensures that warehouse area constraint should be satisfied for each period.

6.3 GENETIC ALGORITHM MODELLING

Solution for the above mathematical model for multi objective optimization inventory control problem has been developed based on Genetic Algorithm (GA) approach. Detailed description about GA implementation including selection of initial population, execution of GA operation – selection, cross over and mutation, stopping criteria have been furnished in the chapter 3 on Research Methodology. The GA has been implemented on JAVA platform and the program can be run with Net Beans IDE. In the following sections, computational results have been explained along with the procedure for parameter selection for the developed GA model and effect of the various parameters on the GA solution efficiency.

6.3.1 PARAMETER CALIBRATION

Setting GA parameters including the crossover probability (Pc), the mutation probability (Pm), population size (PS), and number of generation (NG) is very important in determining the efficiency of the meta-heuristic algorithms like GA. Different parameter influence on the performance of meta heuristic algorithm is usually investigated with full factorial experiment and exhaustive approach (Montgomery D.C, 2000). However, this approach becomes inefficient when the number of factors becomes significantly high and it will be difficult to carry out experiments for all the possible combination of influencing factors. In such situations, practical solution is suggested by fractional factorial experiments (FFEs) which will reduce the number of required tests. However, FFEs only allow a part of total possible combinations to estimate the main effect of factors and some of their interactions.

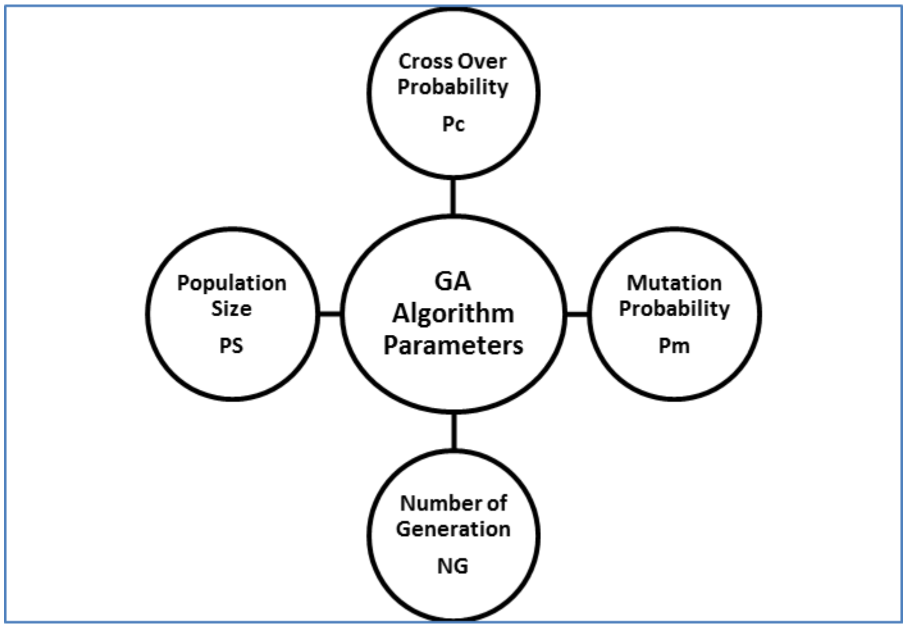


Fig 6.1 GA algorithm parameters (Javed Sadegi et al. 2015).

In famous Taguchi optimization technique, FFE implemented by orthogonal arrays are used to study a large number of decision variables with a small number of experiments, thereby building robustness into experimental setup. Taguchi experimental design has been recognized as a cost-effective and labor-saving method that can simultaneously scrutinize several factors and distinguish quickly the factors with principal impacts on final solution. Under Taguchi Method, the factors are separated into two main groups: controllable and noise factors. Noise factors are those over which we have no direct control. Since elimination of the noise factors is impractical and often impossible, the Taguchi method optimizes by minimizing the effect of noise and by determining the optimal level of important controllable factors based on the concept of robustness.

Taguchi uses the concept of Signal/Noise ratio. As already explained under section 3.11.1, the

$$\frac{S}{N} = -10\, log_{10}(Objective\ function)^2 \qquad \text{(Syed, M.M. et al. 2013)}$$

is used for almost all inventory management problems, which are mostly ‘smaller the better’ type.

GA parameters and their level to be considered in Taguchi analysis are listed in Table 6.1

Table 6.1 GA parameters and levels for Taguchi Design

Levels	**GA PARAMETERS**			
	Cross over Probability (Pc)	Mutation probability (Pm)	Population size (PS)	Number of Generations (NG)
1	0.7	0.15	4000	500
2	0.8	0.20	5000	600
3	0.9	0.30	6000	700

Values of 0.7, 0.8 & 0.9 are considered for cross over probability Pc. For mutation probability Pm, 3 levels of 0.15, 0.2 and 0.3 are considered.

For population size and number of generations, the values of 4000, 5000, 6000 and 500, 600, 700 are considered for Taguchi analysis. Minitab 15 is used to employ the Taguchi method. Under the menu options of Minitab, Stat-DOE-Taguchi Design-Create Design is selected. For 3 levels of 4 factors, L9 orthogonal array is suggested which will facilitate representative Fractional Factorial Experimentation. L9 orthogonal array lists the different combinations of factors at different levels at which the response value of experiments have to be determined. 3 items, 3 periods and 3 price breaks data is selected for experimentation and response in terms of minimum total cost is tabulated as shown in Table 6.1. For different combinations of the factor levels, each example is solved three times and the mean response was used in the analysis. Fig. 6.2 shows the main effect plot of SN ratio and Fig 6.3 shows the main effect plot of means for different parameter levels of the proposed algorithms.

The S/N ratio indicates the amount of variation present in the response variable, and the aim is to maximize it. Pc value of 0.9, Pm value of 0.3, PS value of 5000 and NG value of 500 yields the maximum value of S/N ratio as can be seen from Fig. 6.2. This can also be verified from Fig 6.3, where the above values give the best or the lowest value of objective function. Optimal parameter values of the algorithms are shown in Table 6.3.

Table 6.2 Experimental response for Taguchi design

Cross over Probability (Pc)	Mutation probability (Pm)	Population size (PS)	Number of Generations (NG)	Response value obj. function	S/N ratio
0.7	0.15	4000	500	8070423	-138.138
0.7	0.20	5000	600	8071445	-138.139
0.7	0.30	6000	700	8070334	-138.138
0.8	0.15	5000	700	8070656	-138.138
0.8	0.20	6000	500	8072345	-138.140
0.8	0.30	4000	600	8074558	-138.142
0.9	0.15	6000	600	8066732	-138.134
0.9	0.20	4000	700	8065550	-138.133
0.9	0.30	5000	500	8058597	-138.125

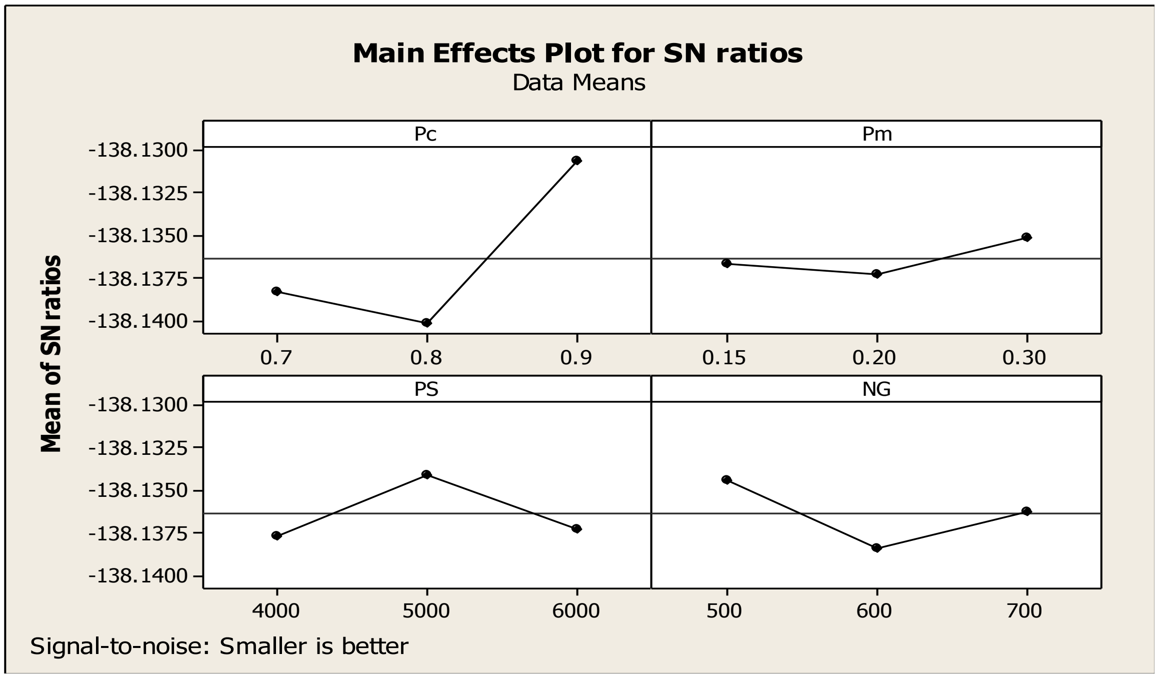


Fig. 6.2 Main effect plot of SN ratios- Taguchi design

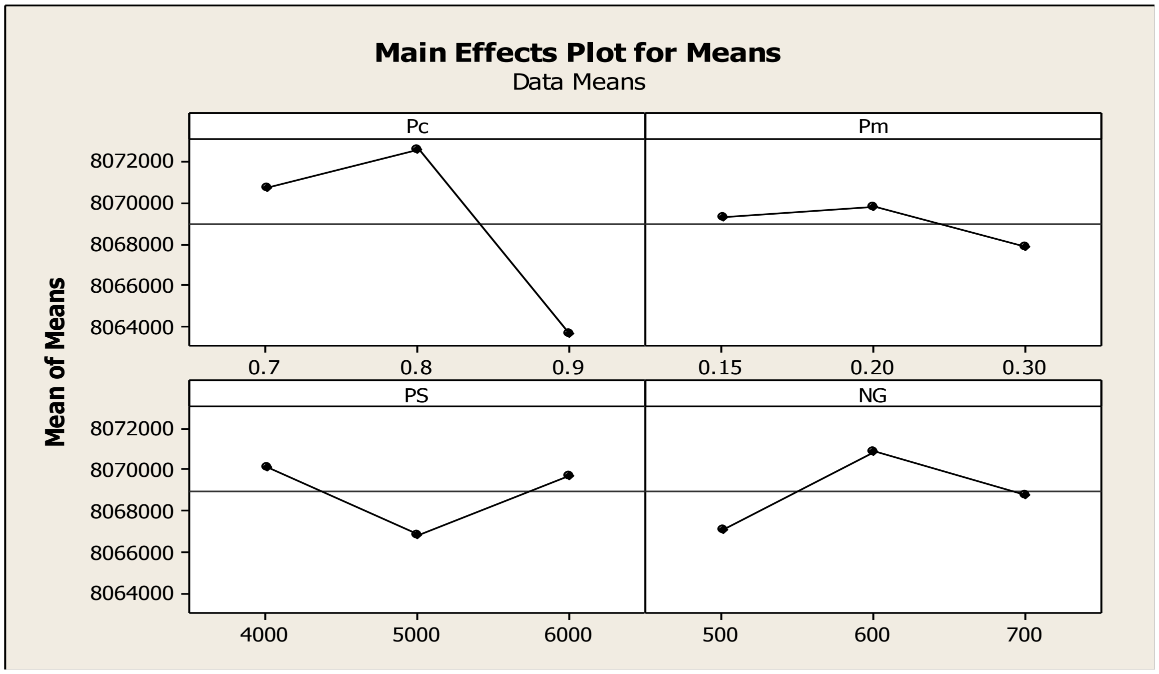


Fig.6.3 Main effect plot of means- Taguchi design

Pc : Cross over Probability   Pm : Mutation probability   PS: Population size

NG :Number of Generation

Table 6.3 Optimum GA parameters

OPTIMUM GA PARAMETERS			
Cross over Probability (Pc)	Mutation probability (Pm)	Population size (PS)	Number of Generations (NG)
0.9	0.30	5000	500

6.3.2 EFFECT OF VARIATION OF GA PARAMETERS ON OBJECTIVE FUNCTION VALUE

Following table 6.4 and Fig 6.4 shows the effect of variation of no. of generations on the objective function value and CPU time. Table 6.4 records the number of generations, objective function value and CPU execution time. Fig 6.4 is the plot of objective function value on the Y-axis against number of generations on X- axis. Keeping Cross over probability Pc, mutation probability Pm and population size PS at optimum values of 0.9, 0.3 & 5000, the number of generations is varied from 50 to 800 in steps and objective function value and CPU execution value for representative problem no. 6 (4-3-3) is tabulated. It shows that the objective function value converges at 500 generations. After 500 generations, there is no much change in objective function value, only CPU time of execution increases.

Similarly, Table 6.5 and Fig 6.5 show the effect of variation of population size on the objective function value and CPU time of execution. GA run carried out for the problem no. 6 (4-3-3) for a wide range of population size keeping other parameters at the optimum value as decided by Taguchi Design. It shows that the objective function value converges at population size of 5000 generations. After 5000, there is no much change in objective function value, only CPU time of execution increases

Table: 6.4 Convergence of objective function value with No. of Generations.

No. of generations	Objective function value (Rs)	Time of execution (Sec)
50	8175155	2
100	8159916	3
200	8105671	6
300	8084691	8
400	8073430	11
500	8065455	14
700	8065445	19
800	8065440	22
1000	8065436	28

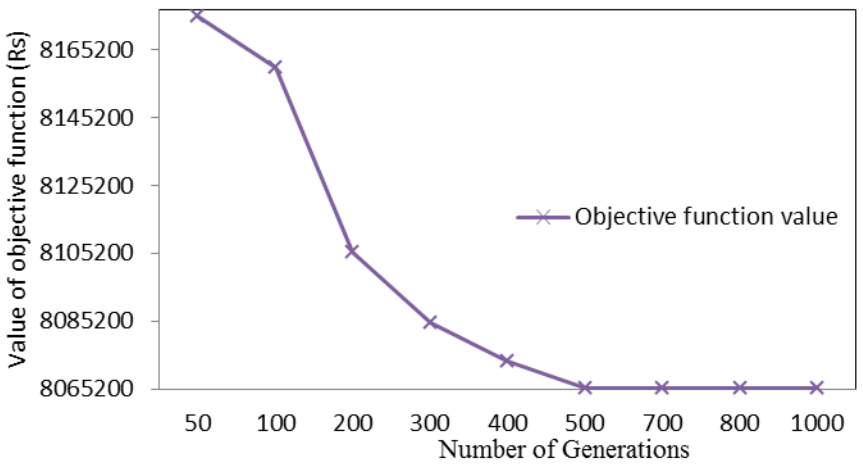


Fig: 6.4 Convergence of Objective function value with No. of Generations- GA model.

Table 6.5 Convergence of Objective Function Value with Population Size.

Population size	Objective function value	Time of execution
10	8123971	1
50	8113395	1
100	8107776	2
300	8094113	2
500	8085252	2
1000	8083290	4
2000	8080560	6
3000	8077696	7
4000	8075565	11
5000	8065560	14
7000	8065520	24
10000	8065510	47

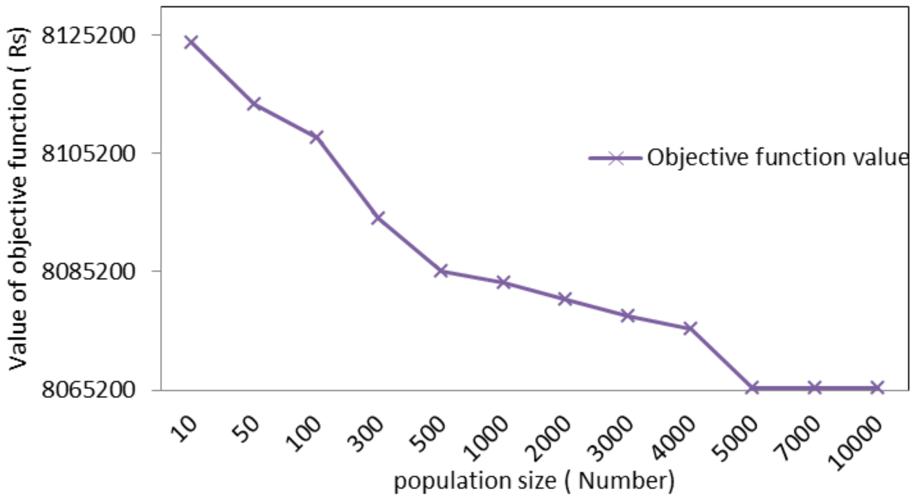


Fig 6.5 Convergence of Objective Function Value with Population Size- GA model.

6.3.3 COMPUTATIONAL RESULTS

This section briefly explains the computational procedure and presents the results of the numerical studies to evaluate the performance and validate the developed GA model. Fig.6.6 shows the GA model performance measures used to establish its validity and compare with ACO model. The best mathematical model will be able to suggest minimum total cost objective function value for the inventory. It will also take minimum computer time to be processed. This is especially important due to the fact that a large number of iterative operations need to be carried out to arrive at a representative value. The best mathematical model would ensure a minimum spread between best and worst solution. Validity of the model can be established by proving that the objective function value obtained is responsive towards the various problem parameters. These four important parameters would constitute a platform to evaluate and compare the performance of GA model as shown in Fig. 6.6.

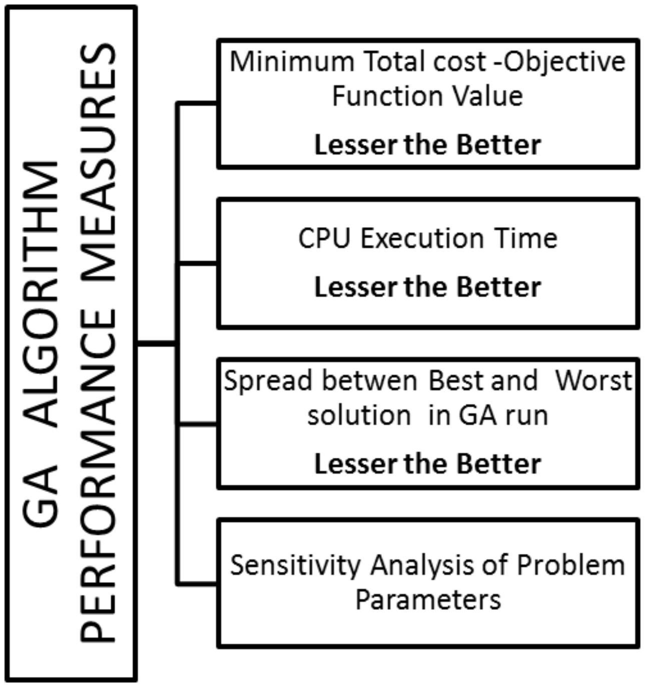

GA ALGORITHM PERFORMANCE MEASURES
Minimum Total cost -Objective Function Value
Lesser the Better
CPU Execution Time
Lesser the Better
Spread betwen Best and Worst solution in GA run
Lesser the Better
Sensitivity Analysis of Problem Parameters

Fig 6.6 GA model performance measure.

Ilkay Saracoglu et al. (2014) adopted the above approach to evaluate the performance of Genetic Algorithm solution proposed by them for multi product multi period continuous review inventory model. They compared the objective function value and CPU execution time of the Genetic algorithm with the Integer Linear Programming solution presented by them.

Javad Sadeghi et al. (2015) implemented the approach of comparing the objective function value and CPU execution time to verify and validate the performance of their Non sorting Genetic Algoritm model in the case of multi echelon vendor managed inventory control.

In order to demonstrate the application of the developed GA, and investigate their performance in suggesting the most economical periodical lot size taking into consideration dynamic demand, and ordering, holding and purchase cost, 8 numerical examples are considered. The data input is from the real industry scenario under study. Data for six most important valve configuration is studied in depth and tabulated as in the tables no 6.6 to 6.8. For ease of identification, the data sets are numbered from 1to 8. Each data set can be conveniently represented as data set no. (c-p-w) where c is the number of items for procurement, p is the period and w is the number of price breaks for discount.

The data presented in Tables 6.4 to 6.8 contain information related to projected bimonthly demand for the different products as predicted from ANN models, ordering cost, inventory carrying cost, purchase cost, reserve stock, warehouse and budget constraints, price breaks and discounts. GA was run 4 times for each problem configuration. For each GA run, the order quantities of each item or product for each period are tabulated. Minimum total cost or the value of objective function which is an important performance measure is listed. The CPU execution time is also noted for each GA run. The spread between best and worst solution is also marked for each GA run. The lesser the spread, better is the algorithm. All the test problems are solved on a lap top with Intel core i3-2100 processor having 3.10 GHz CPU and 4 GB RAM.

Table 6.6 Input data for Lot Sizing Optimisation: Data set no. 1,2,3,4&5.

Data set no. 1 c=1, p=3, w=2	
C	1
p	3
w	2
R11	40
R12	70
R13	50
G1	25000
H1	1020
C11	6025
C12	5723.75
S1	20
A1	5000000
A2	5000000
A3	5000000
m1	0.25
M	300

Data set no.2 c=1, p=3, w=3	
C	1
p	3
w	3
R11	40
R12	70
R13	50
G1	25000
H1	1020
C11	6025
C12	5723.75
C13	5422.5
S1	20
A1	5000000
A2	5000000
A3	5000000
m1	0.25
M	300

Dataset no.3, c=2 p=3, w=2	
c	2
p	3
w	2
R11	40
R12	70
R13	50
R21	130
R22	110
R23	140
G1	25000
G2	20000
H1	1020
H2	500
C11	6025
C12	5723.75
C21	5100
C22	4845
S1	20
S2	40
A1	5000000
A2	5000000
A3	5000000
m1	0.25
m2	0.2
M	300

Data set no. 4 c=2, p=3, w=3	
c	2
p	3
w	3
R11	40
R12	70
R13	50
R21	130
R22	110
R23	140
G1	25000
G2	20000
H1	1020
H2	500
C11	6025
C12	5723.75
C13	5422.5
C21	5100
C22	4845
C23	4590
S1	20
S2	40
A1	5000000
A2	5000000
A3	5000000
m1	0.25
m2	0.2
M	300

Data set no. 5 c=3, p=3, w=3	
c	3
p	3
w	3
R11	40
R12	70
R13	50
R21	130
R22	110
R23	140
R31	300
R32	260
R33	280
G1	25000
G2	20000
G3	9000
H1	1020
H2	500
H3	425
C11	6025
C12	5723.75
C13	5422.5
C21	5100
C22	4845
C23	4590
C31	4575
C32	4346.25
C33	4117.5
S1	20
S2	40
S3	85
A1	5000000
A2	5000000
A3	5000000
m1	0.25
m2	0.2
m3	0.15
M	300

(c= no. of items, p= no. of periods, w= no. of price breaks)

Table 6.7 Input data for Lot Sizing Optimisation: Data set no. 6 & 7.

Data set no.6 c=4, p=3, w=3			
c	**4**	C31	4575
p	**3**	C32	4346.25
w	**3**	C33	4117.5
R11	40	C41	3550
R12	70	C42	3372.5
R13	50	C43	3195
R21	130	S1	20
R22	110	S2	40
R23	140	S3	85
R31	300	S4	30
R32	260	A1	5000000
R33	280	A2	5000000
R41	80	A3	5000000
R42	92	m1	0.25
R43	98	m2	0.2
G1	25000	m3	0.15
G2	20000	m4	0.1
G3	9000	M	300
G4	5000		
H1	1020		
H2	500		
H3	425		
H4	300		
C11	6025		
C12	5723.75		
C13	5422.5		
C21	5100		
C22	4845		
C23	4590		

Data set no.7 c=5, p=3, w=3			
c	**5**	C12	5422.5
p	**3**	C13	5121.25
w	**3**	C21	5100
R11	40	C22	4590
R12	70	C23	4335
R13	50	C31	4575
R21	130	C32	4346.25
R22	110	C33	4117.5
R23	140	C41	3550
R31	300	C42	3195
R32	260	C43	3017
R33	280	C51	4500
R41	80	C52	4275
R42	92	C53	4050
R43	98	S1	20
R51	600	S2	40
R52	665	S3	85
R53	620	S4	30
G1	25000	S5	190
G2	20000	A1	10000000
G3	9000	A2	10000000
G4	5000	A3	10000000
G5	7000	m1	0.25
H1	600	m2	0.2
H2	300	m3	0.15
H3	425	m4	0.1
H4	200	m5	0.1
H5	150	M	600
C11	6025		

Table 6.8 Input data for Lot Sizing Optimisation Data set no. 8.

Data set no.8 c=5, p=3, w=3					
c	6	H6	300	m6	0.15
p	3	C11	6025	M	600
w	3	C12	5422.5		
R11	40	C13	5121.25		
R12	70	C21	5100		
R13	50	C22	4590		
R21	130	C23	4335		
R22	110	C32	4346.25		
R23	140	C33	4117.5		
R31	300	C41	3550		
R32	260	C42	3195		
R33	280	C43	3017		
R41	80	C51	4500		
R42	92	C52	4275		
R43	98	C53	4050		
R51	600	C61	7000		
R52	665	C62	6650		
R53	620	C63	6300		
R61	425	S1	20		
R62	440	S2	40		
R63	460	S3	85		
G1	25000	S4	30		
G2	20000	S5	190		
G3	9000	S6	130		
G4	5000	A1	10000000		
G5	7000	A2	10000000		
G6	7000	A3	10000000		
H1	600	m1	0.25		
H2	300	m2	0.2		
H3	425	m3	0.15		
H4	200	m4	0.1		
H5	150	m5	0.1		

.

6.3.3.1 Results Analysis

Table 6.9 shows the GA run results for data set no.1, where one product, 3 periods and 2 price breaks are considered. Recorded values for order quantities are 51, 59 and 50. The objective function value is Rs 1159882/- and CPU execution time is 11 secs. Table 6.10 shows the results of GA run for data set no. 2, where 1 component, 3 periods and 3 price break point are considered for lot sizing. The results indicate the order quantities of 40, 120 and zero for the three periods. The minimum total inventory cost comes to 1135500/- and computer time of execution is 12 secs.

Table 6.11 & 6.12 tabulates the results for 2 products. GA is run four times. But same result is obtained all the 4 times. GA pre-scribed the same optimal solution each time, the same solution that was obtained via exact methods using the LINDO 14 Linear Integer Programming software. Since this is a case of simple problem which involves small number of variables, the solution obtained using GA is the exact or the most optimal one. Solving the above problem using other optimization softwares like GAMS or LINDO also gives the same result. This fact validates the model. Having verified its ability to optimize the lot sizing, GA can now be used to solve the complex problems having more number of decision variables. Comparison between the results of the GA and LINGO, for small-size problems, shows that we can also trust the GA for the larger problem sizes.

This approach for validation of population based meta heuristic algorithm was adopted by Hamed Soleimani et al. (2015), Fardin et al. (2015), Kuo, R.J et al. (2014) in their research work in the field of Supply chain management and Inventory control. Smaller problem instances having limited number of variables were solved using the exact method. Proposed heuristic algorithms were validated by comparing the solutions obtained from them with the solutions from exact method.

When only one discount is considered the order quantity lot was 50, 59 & 50, respectively, for the 3 periods, as shown in Table 6.9. The minimum total cost was Rs 1159882/-. With 2 discounts or 3 price breaks as in data set no. 2, the most economical lot was found to be 40, 120, 0 respectively for 3 periods as can be seen from the table 6.10. The minimum total cost also came down to Rs 1135500/- due to savings from one less order and higher discount above the price break quantity 100 units, in spite of higher carrying cost expenses.

For the lot sizing decision problems for 3 components and above, the number of decision problems increases and we can find different solutions in different GA runs. This is evident from Table 6.13 to 6.16. Four problems are considered with same number of periods and price breaks but different number of items. All the 4 tables show the GA run results in terms of order quantities and optimal total cost and CPU processing time.

The average optimal cost for the data set no. 5 (3-3-3), data set no. 6 (4-3-3), data set no. 7 (5-3-3) and data set no. 8 (6-3-3) are Rs 7016587/-, Rs 8068146/-, Rs15702229/- and Rs25457852/-, respectively, as seen from Tables 6.13 to 6.16 . The CPU time also increases from 8 sec for 3 items to 15 sec for 6 items. In section 6, the objective function value and CPU time from different optimization models will be compared. Table 6.17 summarizes the lot sizing optimization results for different data sets.

Table 6.9 GA run for Multi item Multi period Lot size Optimisation Data set no. 1: c=1, p=3, w=2.

Run no.	Item no.	Order Qty. (No) Period 1	Order Qty (No) Period 2	Order Qty (No) Period3	Best Sol Total Cost (Rs)	Spread bet. Best and worst solution (Rs)	CPU time (Sec)
1	1	51	59	50	1159882	144660	11
2	1	51	59	50	1159882	144660	11
3	1	51	59	50	1159882	144660	11
4	1	51	59	50	1159882	144660	11

(c= no. of items, p= no. of periods, w= no. of price breaks).

Table 6.10 GA run for Multi item Multi period Lot size Optimisation Data set no. 2: c=1, p=3, w=3.

Run no.	Item no.	Order Qty. (No) Period 1	Order Qty (No) Period 2	Order Qty (No) Period3	Best sol total Cost (Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	1	40	120	0	1135500	121445	11
2	1	40	120	0	1135500	121445	11
3	1	40	120	0	1135500	121445	11
4	1	40	120	0	1135500	121445	11

Table 6.11 GA run for Multi item Multi period Lot size Optimisation:Data set no. 3: c=2, p=3, w=2.

Run no.	Item no.	Order Qty. (No) Period 1	Order Qty (No) Period 2	Order Qty (No) Period3	Best sol total Cost(Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	1	51	59	50	3215982	336700	12
	2	130	110	140			
2	1	51	59	50	3215982	336700	12
	2	130	110	140			
3	1	51	59	50	3215982	336700	12
	2	130	110	140			
4	1	51	59	50	3215982	336700	12
	2	130	110	140			

(c= no. of items, p= no. of periods, w= no. of price breaks).

Table 6.12 GA run for Multi item Multi period Lot size Optimisation:Data set no. 4: c=2, p=3, w=3.

Run no.	Item no.	Order Qty. (No) Period 1	Order Qty (No) Period 2	Order Qty (No) Period3	Best sol total Cost (Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	1	40	120	0	3494700	316015	12
	2	130	110	140			
2	1	40	120	0	3494700	316015	12
	2	130	110	140			
3	1	40	120	0	3494700	316015	12
	2	130	110	140			
4	1	40	120	0	3494700	316015	12
	2	130	110	140			

Table 6.13 GA run for Multi item Multi period Lot size Optimisation Data set no. 5: c=3, p=3, w=3.

Run no.	Item no.	Order Qty. (No) Period 1	Order Qty. (No) Period 2	Order Qty. (No) Period3	Best Sol Total Cost (Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	1	42	118	131	7015841	493352	13
	2	131	120	129			
	3	361	208	271			
2	1	40	120	0	7010875	497825	13
	2	134	108	138			
	3	372	468	00			
3	1	43	117	0	7017807.	485675	13
	2	138	102	140			
	3	372	468	0			
4	1	41	119	0	7021825	491372	13
	2	130	110	140			
	3	377	426	377			
Average Objective Fn value & Avg. spread					7016587	492056	13

(c= no. of items, p= no. of periods, w= no. of price breaks).

Table 6.14 GA run for Multi item Multi period Lot size Optimisation Data set no. 6: c=4, p=3, w=3.

Run no.	Item no.	Order Qty. (No) Period 1	Order Qty (No) Period 2	Order Qty (No) Period3	Best sol total Cost(Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	1	51	109	0	8090702	510110	12
	2	131	109	140			
	3	367	227	246			
	4	80	106	84			
2	1	41	119	0	8058597	514989	12
	2	140	106	164			
	3	368	472	0			
	4	81	96	93			
3	1	40	120	0	8067023	514867	12
	2	130	113	137			
	3	367	228	245			
	4	80	119	79			
4	1	40	120	0	8056263	498388	**12**
	2	130	112	138			
	3	363	221	256			
	4	83	89	98			
Average Objective Fn value & Avg. spread					8068146	509588	**12**

(c= no. of items, p= no. of periods, w= no. of price breaks).

Table 6.17 lists the different data sets considered for multi item multi period dynamic lot sizing inventory management problem differentiated by the number of items , periods and price breaks considered. The solution set for each of data set GA run suggests the order quantities for the different periods.

Table 6.15 GA run for Multi item Multi period Lot size Optimisation Data set no. 7: c=5, p=3, w=3.

Run no.	Item no.	Order Qty. (No) Period 1	Order Qty (No) Period 2	Order Qty (No) Period3	Best sol. total Cost(Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	1	57	103	0	15764097	802316	12
	2	168	212	0			
	3	407	395	38			
	4	270	0	0			
	5	634	742	509			
2	1	45	115	0	15659722	828462	12
	2	153	227	0			
	3	367	208	265			
	4	120	150	0			
	5	674	623	588			
3	1	47	113	0	15736621	785438	12
	2	164	216	0			
	3	312	254	274			
	4	270	0	0			
	5	828	446	611			
4	1	48	112	0	15648478	790905	12
	2	165	215	0			
	3	369	407	64			
	4	270	0	0			
	5	600	665	620			
Average Objective Fn value & Avg. spread					15702229	801780	12

(c= no. of items, p= no. of periods, w= no. of price breaks).

Table 6.16 GA run for Multi item Multi period Lot size Optimisation Data set no. 8: c=6, p=3, w=3.

Run no.	Item no.	Order Qty. (No) Period 1	Order Qty. (No) Period 2	Order Qty (No) Period3	Best sol total Cost (Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	1	105	55	0	25260387	675846	18
	2	170	210	0			
	3	364	209	267			
	4	270	0	0			
	5	620	651	614			
	6	426	442	457			
2	1	54	106	0	25287605	692205	18
	2	159	221	0			
	3	370	214	256			
	4	149	93	28			
	5	603	662	620			
	6	425	440	460			
3	1	57	103	0	25307731.25	699552	13
	2	157	223	0			
	3	430	387	23			
	4	150	120	0			
	5	617	650	618			
	6	428	438	459			
4	1	81	79	0	25305686	658103	13
	2	167	213	0			
	3	373	223	244			
	4	118	137	15			
	5	602	665	618			
	6	428	437	460			
Average Objective Fn value & Avg. spread					25290352	681426.	15.5

(c= no. of items, p= no. of periods, w= no. of price breaks).

Table 6.17 Performance Parameters of GA for different data sets.

Data set no.	Data set Description	Best Sol Total Cost (Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	c=1, p=3, w=2	1159882	144660	11
2	c=1, p=3, w=3	1135500	121445	11
3	c=2, p=3, w=2	3215982	336700	12
4	c=2, p=3, w=3	3494700	316015	12
5	c=3, p=3, w=3	7016587	492056	11
6	c=4, p=3, w=3	8068146	509588	12
7	c=5, p=3, w=3	15702229	801780	12
8	c=6, p=3, w=3	25290352	681426	13

(c= no. of items, p= no. of periods, w= no. of price breaks).

When there are six items, (c=6), for each of six component, the solution suggests order quantities for each period so that total inventory cost is minimsed considering the ordering cost, holding cost and purchase cost subject to the constraint of maximum budget and ware house area. The optimum objective function value or the minimum inventory cost in each of the data set scenario is listed along with CPU time for algorithm execution.

6.4 SENSITIVITY ANALYSIS

Sensitivity analysis is the key to Performance appraisal of the mathematical models and their solutions (Devendra Choudhary et al. 2011). Various research works in the field of supply chain and inventory management point to the application of sensitivity analysis of the problem variables as a proven approach to validate their mathematical model[Hamed Soleimani et al. 2015, Ata Allah Taleizadeh et al. 2013, Gupta, R. K et al. 2009].

In this section, the effect of variation in problem parameters is investigated on multi-period lot-sizing decision from our algorithm. The ordering cost, holding cost, purchase cost in terms of discounts and price breaks are the major input parameters for the proposed algorithm.

6.4.1 Sensitivity to ordering cost

Data set no. 1 (c=1, p=3, w=2) is used to study the effect of ordering cost on the procurement lot size and total cost by keeping all the data same as in table 6.6, except the ordering cost Gu. Table 6.18 lists the different periodic lot size by running the GA by varying the ordering cost. Fig 6.7 graphically represents the variation in suggested order quantities with the variation in ordering cost through bar chart.

Table 6.18 Ordering cost –sensitivity analysis- GA model.

Run no.	Ordering cost(Rs)	Order Qty. (No) Period 1	Order Qty (No.) Period 2	Order Qty.(No) Period 3	Best Sol Total Cost (Rs)
1	25000	51	59	50	1159882
2	35000	51	59	50	1189882
3	35940	51	109	0	1192700
4	111180	160	0	0	1343180

It can be seen from the Table 6.18 and Fig 6.7 that the algorithm suggested order quantities for 3 periods when the ordering cost was less than around Rs. 35940/- . After that the lot sizing suggested to buy the quantities only at 2 periods reducing the number of annual order to 2. The saving in ordering cost was more than the increase in holding cost expenses.

When the ordering cost is still higher at Rs 111180/-, the algorithm suggests only one lot to be bought at the year beginning on the basis of ordering cost- holding cost trade off. This is the clear proof to show that the proposed GA is responsive to the problem parameter ordering cost and suggested order quantities closely follow the cost trade offs to give minimum inventory cost.

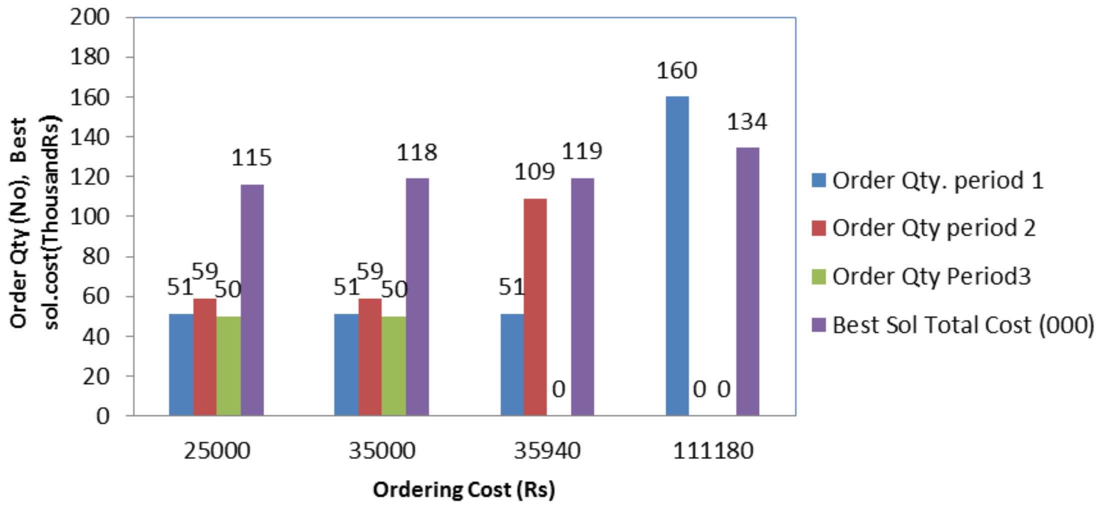


Fig 6.7 Ordering cost: Sensitivity Analysis- GA model

Fig. 6.8 is the plot of order quantities on Y axis against holding cost on X-axis which can be used to clearly visualize the sensitivity of holding cost on order quantities. For the present problem, at holding cost of Rs1020/-, the order quantities are 51, 59 and 50 for the three periods. If the holding cost is increased, a point will be reached at around Rs 1100/-, where the GA algorithm will suggest the JIT(Just in Time) procurement of quantities 40, 70, 50, respectively, for the 3 periods because at this cost, there can not be any trade off between holding and ordering cost.

Table 6.19 Holding Cost –sensitivity analysis- GA model

Run no.	Ordering cost(Rs)	Order Qty. (No) period 1	Order Qty (No.) period 2	Order Qty.(No) Period3	Best Sol Total Cost(Rs)
1	1020	51	59	50	1159882
2	1100	40	70	50	1171912
3	800	51	109	0	1126600
4	225	160	0	0	1010550

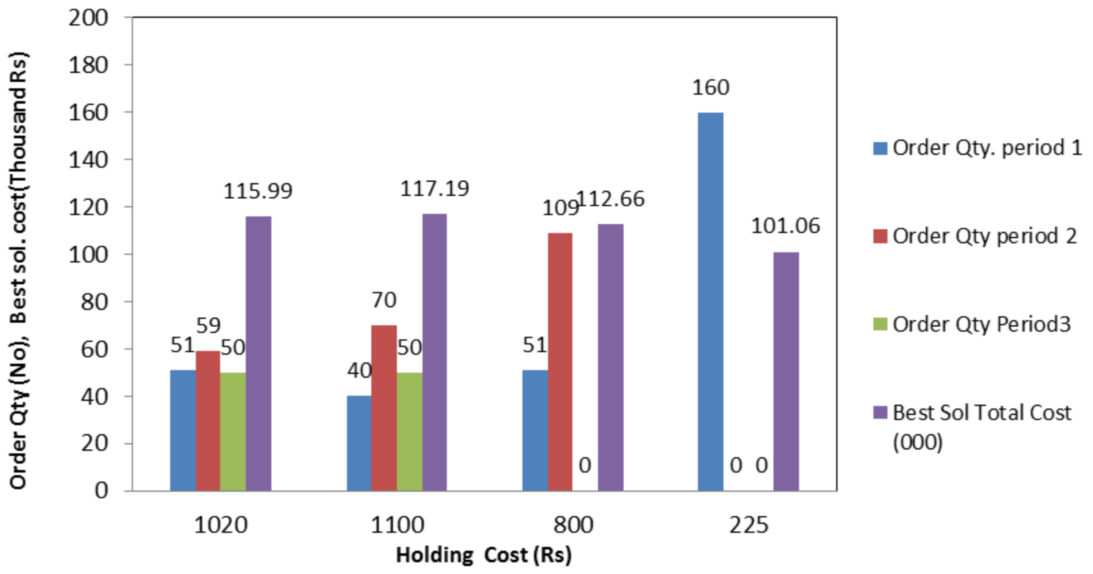


Fig 6.8 Holding Cost –sensitivity analysis-GA model

It will be interesting to note that as the holding cost is reduced to around Rs, 800/-, the first trade off point between ordering and holding cost is reached. So, it becomes more advantageous to reduce one order in spite of higher carrying expenses due to larger inventory. So, optimized order quantities are obtained as 51, 109, 0. When the holding cost is further reduced, at around Rs 220/-, it becomes less expensive to buy all the annual demand of 160 at the beginning of the year, due to saving in ordering cost. The above analysis highlights the sensitivity of decision variable holding cost on order quantities suggested by the proposed GA model.

6.4.3. Sensitivity to discounts and price breaks

Referring to Table 6.6, data set no. 1 and 2 are same except that in data set no. 2, there is an additional price break that is specified at quantity 100 which will become eligible for discount of 10%. Comparing GA solution to data set 1 and 2, (Table 6.9 & 6.10), it is clear that order quantity has been increased to full requirement of 120 units in the second period, as there is a cost reduction due to higher discounts of 10% at the price break quantity 100 units. Additional sensitivity exercises are carried out as per the following

Table 6.20 taking the data from data set no.2 , but varying the discounts and price break quantities.

Table 6.20 Sensitivity to discounts and price breaks- GA model.

No.	Discount/price break	Discount/price break 2	Order Qty.(No) Period 1	Order Qty (No) Period 2	Order Qty (No) Period3	Best Sol Total Cost(Rs)
1	5% at & above 50 Qty	NIL	51	59	50	1159882
2	5% at & above 60	10% @ and above 100	40	120	0	1135500
3	25% at and above 150	Nil	160	0	0	1064200

In the scenario no.1, all the order quantities are selected so that they are above 50 to see that they are eligible for the 5% discount at the first price break. In the scenario 3, there is single discount of 25% above price break quantity of 150. To avail this huge discount, the order quantity of 160 is suggested in the first period itself to ensure that the order quantity stays at above 150.

It is evident from above three sub sections, that procurement lot-sizing model proposed is sensitive to the variation in problem parameters. The computational results suggest that the proposed model captures all realistic constraints in multi-period procurement lot-sizing decision making process and analyzes tradeoffs in cost objectives. Optimal procurement lot-size is obtained by striking best tradeoffs among multiple cost objectives.

Smaller lot-size reduces inventory holding cost but increases purchasing cost and ordering cost due to lack of economy of scale. Larger lot-size reduces purchasing cost, and leads to higher inventory holding cost. All these are reflected in our sensitivity analysis.

6.5 COMPARISION OF PERFORMANCE OF ACO & GA MODEL

It is evident from the above computation results that both the models are robust enough and can be relied upon to give most economical lot sizes for periodic reorder of the multi item multi period procurement. Two methods are compared on the basis of their performance parameters as shown in Table 6.21 based on the result of problem solutions.

Table 6.21 Comparison of performance parameters of ACO and GA model.

Data set no.	Data set Description	GA model			ACO model		
		Best sol. Total Cost (Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)	Best sol.Total Cost (Rs)	Spread bet. best and worst solution (Rs)	CPU time (Sec)
1	c=1, p=3, w=2	1159882	144660	11	1159882	144660	11
2	c=1, p=3, w=3	1135500	121445	11	1135500	121445	12
3	c=2, p=3, w=2	3215982	336700	12	3215982	336700	20
4	c=2, p=3, w=3	3494700	316015	12	3494700	316015	22
5	c=3, p=3, w=3	7016587	492056	11	7016587	492056	40
6	c=4, p=3, w=3	8068146	509588	11	8082410	483039	62
7	c=5, p=3, w=3	15702229	801780	12	15899639	747235	75
8	c=6, p=3, w=3	25290352	681426	13	25624848	650104	80

(c= no. of items, p= no. of periods, w= no. of price breaks).

One way ANOVA is employed to compare statistically the performance of GA and ACO solution methodologies. Minitab 15 software is used to execute the ANOVA. Three different tests were carried out to compare the performance parameters of best solution cost, CPU time and spread between best and worst solution. The output is shown in Table 6.22, 6.23 and 6.24. The results are discussed in each of the following sections.

6.5.1 Comparison of best solution total cost or objective function

It can be observed from Table 6.20 and Fig. 6.12 that the best total cost or the value of objective function is almost same with GA and ACO and there is no significant change. For initial problems where the number of variables is smaller, the objective function values are exactly same. After the data set no. 6, with the increase in the number of items c in the problem definition, the number of variables to be optimized increases. GA was able to give small improvement in the total cost of inventory investment when compared to ACO.

Table 6.22 lists the result of ANOVA analysis to compare the solution methodologies statistically on the basis of best solution cost or best fitness value. The output indicates that at confidence level 95%, the two algorithms have no statistically significant differences in the best fitness value between their means as P-value is 0.987 >0.05. This follows from the acceptance of null hypothesis that the two population means are equal.

Table 6.22 ANOVA results to compare best solution cost.

SOURCE	DF	SS	MS	F	P-value
Solution Methodology	1	18643854306	18643854306	0	0.987
Error	14	1.00697E+15	7.19267E+13		
Total	15				

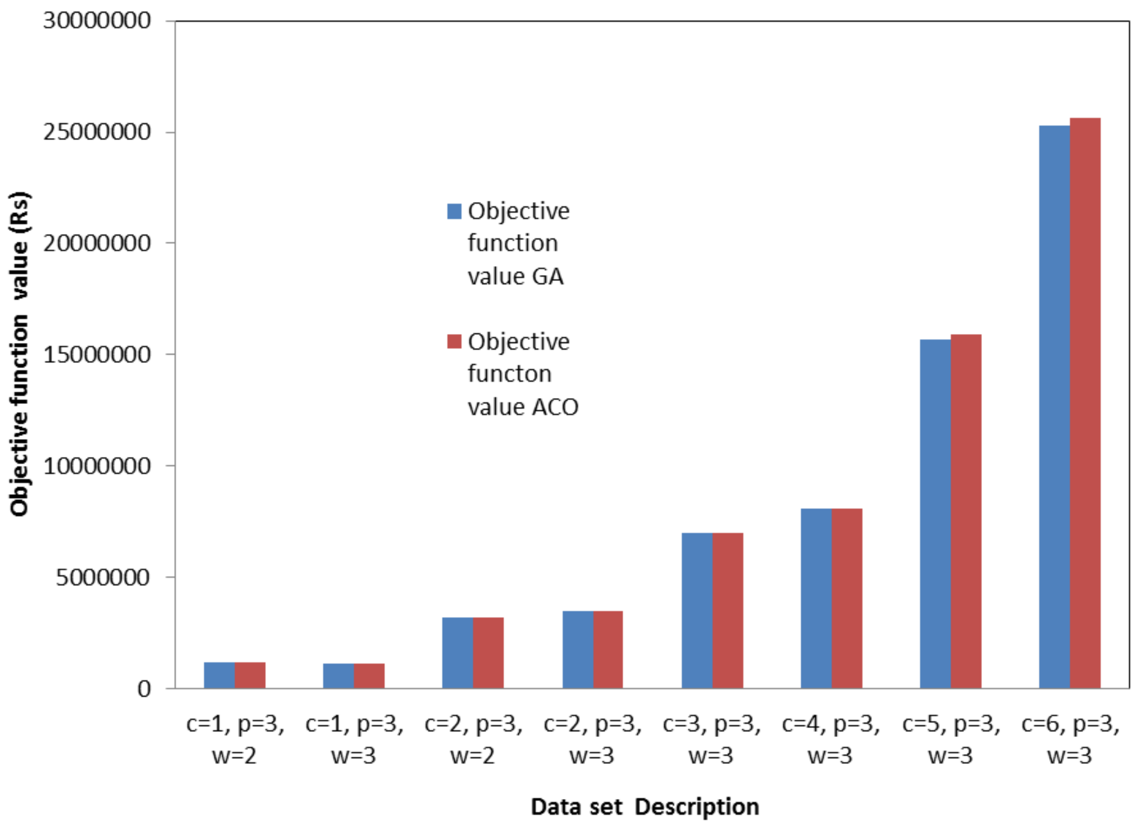


Fig 6.9 Comparison of Objective Function value GA & ACO.

6.5.2 Comparison of CPU time of execution for GA and ACO model

Table 6.23 lists the result of ANOVA analysis to compare the solution methodologies statistically on the basis of CPU time of execution based on the results of 8 problems. The output indicates that at confidence level 95%, the two algorithms have statistically significant differences as far as CPU time is concerned, as P-value is lesser than 0.05. This conclusion follows from rejection of null hypothesis that the two population means are equal.

Fig 6.10 and Table 6.21 illustrates the comparative evaluation of GA & ACO models with respect to the CPU time of execution of the algorithm. It is highly evident that GA is far superior to ACO based on this performance parameter.

Table 6.23 ANOVA results to compare the solution methodologies based on CPU time of execution.

SOURCE	DF	SS	MS	F	P-value
Solution Methodology	1	3306	3306	8.26	0.012
Error	14	5604	400		
Total	15				

With smaller number of variables, there is no much difference in the execution time. But as the problem becomes more complex and the number of variables increases, there is a distinct trend of lesser CPU time in the case of GA model.

This shows that for our mathematical model of inventory cost optimization, GA works more efficiently than ACO in terms of CPU time. This can be attributed to the simplicity of GA compared to ACO in the algorithm formulation which is reflected in the lesser CPU time.

6.5.3 Comparison of Spread between Best and Worst Solution for GA and ACO models

Table 6.20 and Fig 6.11 show the comparison of spread between best and worst solutions for GA and ACO. It can be seen that ACO has got superior performance than GA on this count as spread in the objective function value between the best and worst solution is lesser in case of ACO than in GA.

Table 6.24 lists the result of ANOVA analysis to compare the solution methodologies statistically on the basis of spread between best and worst solution based on the results of 8 problems.

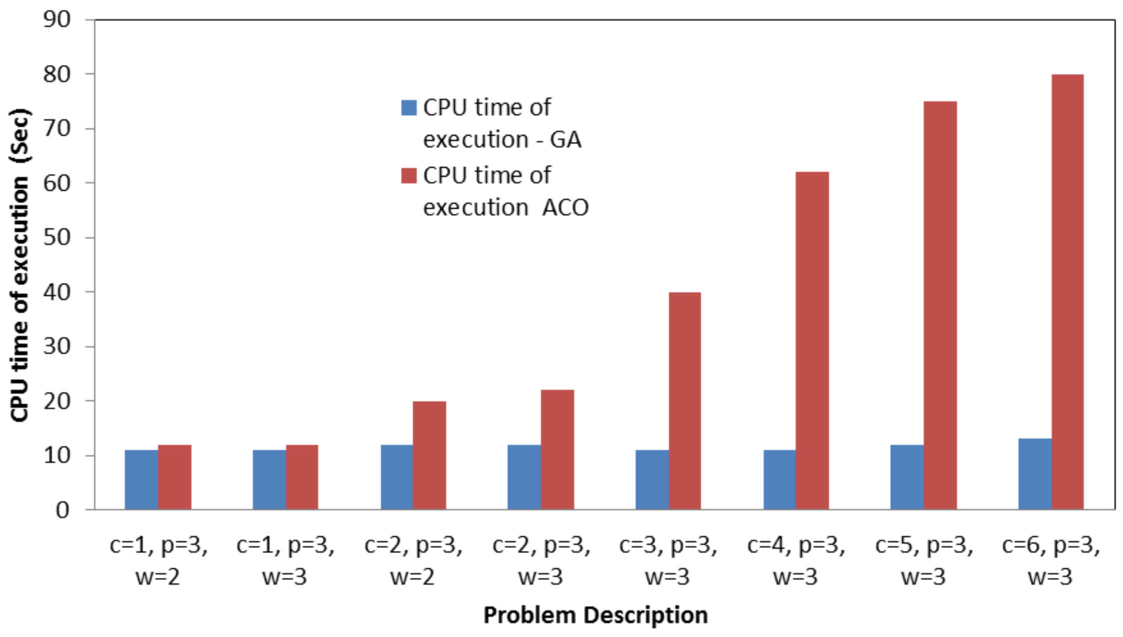


Fig 6.10 Comparison of CPU time of execution for GA and ACO model.

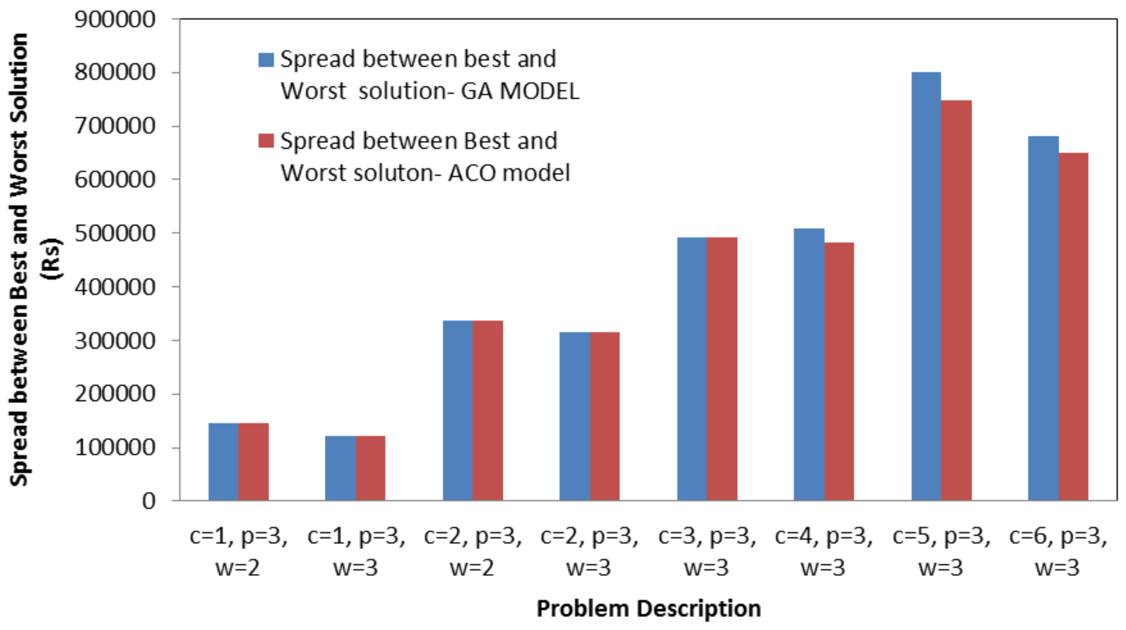


Fig 6.11 Comparison of Spread between Best and Worst Solution for GA and ACO model.

The output indicates that at confidence level 95%, the two algorithms have no statistically significant differences as far as this performance parameter is concerned, as P-value is 0.906 >0.05. This follows from the acceptance of null hypothesis that the two population means are equal.

Table 6.24 ANOVA results to compare the solution methodologies based on spread between best and worst solution.

SOURCE	DF	SS	MS	F	P-value
Solution	1	789834816	789834816	0.01	.906
Error	14	789834816	54363070713		
Total	15				

6.5.4 Role of GA and ACO models in decision making

Results obtained from the present research work based ACO and GA model can be utilised as major decision making platform by the company for procurement lot sizing. The company need not depend on the subjective decision of purchase managers regarding the reorder point and reorder quantities. Since all the practical considerations including reserve stock, budget and storage space constraints have been incorporated into the model, the company can effectively use this model for important inventory management decisions on how much to buy and when to buy for its multi item multi period procurement keeping the total inventory cost at the minimum.

6.6 Summary of results:

1. The mathematical model is constructed for inventory management problem for optimizing multi item multi period lot sizing considering deterministic but variable demand. Model is made more realistic to suit the requirements of company under study by constraining budget and storage space.
2. Since the model is NP hard (non-deterministic polynomial-time hard**)**, Meta heuristic ACO algorithm is developed for the solution in the last chapter. Since

there is no bench mark available in the standard to compare the performance of ACO model, another meta heuristic model GA is developed. Achieving improvement in algorithm performance parameters is added objective for development of GA model.

3. GA model parameters are calibrated using Taguchi design of experiments.
4. GA model solution is evaluated based on performance parameters- minimum total cost of best solution which is the objective function value, CPU processing time, spread between best and worst solution & problem parameter sensitivity analysis.
5. GA run carried out for each of 8 problem scenario and performance parameters along with order quantities are tabulated. Simple problems solutions with single item are verified against exact method which validates the proposed GA model
6. Comprehensive graph developed indicating the GA model solution performance parameters for each of the problem scenario.
7. Sensitivity analysis carried out for the problem parameters like ordering cost, holding cost and price break and discount. Cost trade off scenarios are well verified.
8. The computational results suggest that the proposed model captures all realistic constraints in multi-period procurement lot-sizing decision making process and analyzes tradeoffs in cost objectives. The model is robust enough to accommodate the real life scenario in an industrial and trading set up for suggesting the optimum procurement lot sizes.
9. Comparative evaluation of the GA and ACO model reveals that GA is better than ACO as far as CPU time of execution is concerned. There is no significant difference in the objective function value that both these methods can achieve. The spread in objective function value between the best and worst solutions is higher in GA than ACO which shows the superiority of ACO on this count.
10. One way ANOVA analysis results are also used to compare these solution methodologies, GA and ACO. It accepts null hypothesis that the two populations are equal in case of best objective function value and spread between best and

worst objective function value, which means that there is statistically no significant difference. ANOVA rejects null hypothesis in the case of CPU execution time which means that the there is significant difference in the mean value of populations based on this performance parameter. Present research work suggests that for a comparable problem with sufficient complexity, the CPU execution time has been reduced by 400% when using GA model.

7. CONCLUSIONS AND SCOPE FOR FUTURE WORK

In the present research work, the application of AI techniques for the demand forecast and inventory management has been explored. Neural network models with different architecture have been suggested for demand forecast and their prediction accuracy have been determined. MLP architecture has been compared with RBFNN in terms of Mean absolute Error of prediction. In the second part of research work, multi objective optimization model has been developed for multi item multi period procurement lot sizing under determinate but variable demand condition. ACO and GA programs have been developed for solving the mathematical model. The models have been validated with the real time industrial data and their comparative merits and demerits have been studied. Following broad conclusions have been drawn from the present research study.
The novelty of present research work is the integrated AI application on demand forecast and inventory management. There are independent studies using AI techniques for demand forecast [[Aburto et al. 2007, Wong, B. K. et al. 2007 Nikolaos Kourentzes (2013)] and optimum lot sizing [Zhong Yao et al. 2011, Baruch Keren,2009, Chia-Shin Chung et al. 2013]. In this research work, the output of neural network demand forecast is used for lot sizing optimization using GA application, which is a novel approach. Another novelty has been the application of periodic review lot sizing model of inventory management to multi item system. Most of the earlier works have been targeted to single item multi period. The present research work adopts a new approach of using both ACO and GA for the lot sizing optimization of multi item multi period periodic review inventory management which can be considered significant contribution.

Following broad conclusions have been drawn from the present research study.

CONCLUSIONS

1. Neural network can effectively be used for the demand forecast with different network architecture like MLP or RBFNN. Neural network results in higher prediction accuracy than the traditional methods. The ability to increase

forecasting accuracy will result in lower costs and higher customer satisfaction because of more on-time deliveries. The proposed methodology can be considered as a successful decision support tool in forecasting customer demand. RBFNN can be configured with random selection of centers or self organized selection of centers using clustered algorithms like Fuzzy C means. The width can be fixed and equal or variable and determined using P-nearest neighbor heuristic. FCM centers yielded a best prediction accuracy of 4.38% whereas the maximum accuracy in the case of random centers was 4.81%.

2. Meta heuristic algorithms like GA and ACO can effectively be used to solve NP hard (non-deterministic polynomial-time hard**)** mathematical models related to inventory management problems like the multi period multi item periodic lot sizing problem. ACO program parameter can be set by running the representative problem several times and optimizing the objective value function. Simple problem solutions for multi period lot sizing with single item are verified against exact method like LINDO or GAMMAS software which validate the proposed ACO and GA model. Sensitivity analysis which is carried out for both GA and ACO programs with respect to the problem parameters like ordering cost, holding cost and price break and discount, proves that cost trade off scenarios are well verified. Computational results concluded that suggested ACO and GA models properly analyse the trade off in cost objectives and that they capture well, all the realistic constraints in the decision making process for the multi period procurement lot sizing problem. The models are robust enough to accommodate the real life scenario in an industrial and trading set up for suggesting the optimum procurement lot sizes.

3. Comparative evaluation of the GA and ACO model reveals that GA is better than ACO as far as CPU time of execution is concerned. There is no significant difference in the objective function value that both these methods can achieve.

The spread in objective function value between the best and worst solutions is higher in GA than ACO which shows the superiority of ACO on this count, even though the difference in spread is minor and statistically insignificant. One way ANOVA analysis results are also used to compare these solution methodologies, GA and ACO. It accepts null hypothesis that the two populations are equal in case of Best objective function value and spread between best and worst objective function value, which means that there is statistically no significant difference. ANOVA rejects null hypothesis in the case of CPU execution time which means that there is significant difference in the mean value of populations based on this performance parameter. Present research work suggests that for a comparable problem with sufficient complexity, the CPU execution time has been reduced by 400% when using GA model.

7.2 DIRECTIONS FOR FUTURE STUDY

Present research work has revealed that ANN with MLP and RBFNN network gives much higher prediction accuracy when used for demand forecast. Also ACO and GA models can be used for optimizing multi item multi period lot sizing where the exact methods would take very long time for the solution or become unstable due to the large number of decision variable. Even though a huge amount of research work has been done in the field of inventory management and demand forecast, there are still a lot of new avenues that can be explored in the application of AI technique in these respective fields.

- Future research can explore the possibility of using other ANN types like recurrent neural networks to make a similar approach and better the accuracy of prediction.
- Other meta-heuristic search algorithms such as simulated annealing, may be employed for the optimization of inventory planning and procurement lot sizing and a comparison may be made among the algorithms.

- Uncertainty in the estimation of the different variables like carrying cost, ordering cost etc. can be modeled by the fuzziness to take care of their stochastic nature which will give a different approach to problem solution.
- The model can be extended to accommodate some more real world scenario like rejections and late deliveries from vendor side.
- Different variations in Genetic algorithm and ACO can be explored to improve the optimization of the procurement lot sizes.
- Co-ordination between supplier and buyer is an important influencing factor in the recent trends of collaborative procurement strategies. This factor can be modeled in optimizing the multi item multi period lot size.

REFERENCES

Aburto, L., & Weber, R. (2007). “Improved supply chain management based on hybrid demand forecasts.” Applied Soft Computing, 7, 136–144.

Aggarwal, A., & Park, J. K. (1993). “Improved algorithms for economic lot-size problems.” Journal of Operations Research, 45, 49‑71.

Alejandro Serran, Rogelio Oliva, Santiago Kraiselbur (2017). “On the cost of capital in inventory models with deterministic demand.” International Journal of Production Economics, 15, 14-20.

Ali Diabat, Rany, Deskoores (2016). “Hybrid genetic algorithm based heuristic for an integrated supply chain problem.” Journal of Manufacturing Systems, 50, 9568-9575.

Ali Roozbeh Nia, Mohammad Hemmati Far, Seyed Taghi Akhavan Niaki (2014). “A fuzzy vendor managed inventory of multi-item economic order quantity model under shortage: An ant colony optimization algorithm.” Int. J.Production Economics, 155, 259–27.

Al-Saba, T. & El-Amin, I. (2009). “Artificial neural networks as applied to long-term demand forecasting.” Artificial Intelligence in Engineering, 13, 189‑197.

Amy H.I. Lee, He-Yau Kang, Chun-Mei Lai, Wan-Yu Hong (2013). “An integrated model for lot sizing with supplier selection and quantity discounts.” Applied Mathematical Modelling, 37, 4733–4746.

Angappa Gunasekaran, Eric W.T.Ngai (2014). “Expert systems and artificial intelligence in the 21st century logistics and supply chain management.” Expert Systems Applications ,Volume 41, Issue 1, Pages 1-4.

Ann M. Noblesse, Robert N. Boute, Marc R. Lambrecht, Benny Van Houdt (2014). “Lot sizing and lead time decisions in production/inventory systems.” International Journal of Production Economics, Volume 155, Pages 351-360

Arindam Roy A, Sova Pal, Manas Kumar Maiti (2009). “A production inventory model with stock dependent demand incorporating learning and inflationary effect in a random planning horizon: A fuzzy genetic algorithm with varying population size approach.” Computers & Industrial Engineering, 57, 1324–1335.

Aris, Syntetos, Zied Babai, John E. Boylan (2016). “Supply chain forecasting: Theory, practice, their gap and the future.” European Journal of Operational Research, 252, 1–26.

Ata Allah Taleizadeh, Seyed Taghi Akhavan Niaki (2013). “A hybrid method of fuzzy simulation and genetic algorithm to optimize constrained inventory control systems with stochastic replenishments and fuzzy demand.” Information Sciences, 220, 425–441.

Azzi, A., Battini, D., Faccio, M., Persona, A. (2014). “Inventory holding costs measurement: A multi-case study.” The International Journal of Logistics Management, 25, 109‑132.

Barbara B.Flynn, SadaoSakakibara, Roger G.Schroeder (1990). “Empirical research methods in operations management.” Journal of Operations Management, Volume 9, Issue 2, April 1990, Pages 250-284.

Baruch Keren (2009). “The single-period inventory problem: Extension to random yield from the perspective of the supply chain.” Omega,Volume 37, Issue 4, Pages 801-810.

Ba-Yi Cheng, Joseph Y.T., Leung (2010). “Integrated scheduling of production and distribution to minimize total cost using an improved ant colony optimization method.” Computers & Industrial Engineering, 83, 217‑225.

Beale & Jackson (2000). “Neural Computing: An introduction.” Institute of Physics Publishing, Bristol and Philadelphia, 13, 370-378.

Beccali, M., Cellura, M., Lo Brano, V. & Marvuglia, A. (2004). “Forecasting daily urban electric load profiles using artificial neural networks.” Energy Conversion and Management, 45, 2879–2900.

Behnam Vahdani, Soltani, M. Yazdani, Meysam Mousavi (2017). “A three level joint location-inventory problem with correlated demand, shortages and periodic review system: Robust meta-heuristics.” Computers & Industrial Engineering, Volume 109, Pages 113-129.

Benton, W. C. (1991). “Quantity discount decisions under conditions of multiple items, multiple suppliers and resource limitation.” International Journal of Production Research, 29, 953–961.

Biswajit Sarkar (2013). “A production-inventory model with probabilistic deterioration in two-echelon supply chain management.” Applied Mathematical Modelling, 37, 3138–3151.

Boylan, J.E., Syntetos, A.A., Karakostas, G.C. (2007). “Classification for forecasting and stock control: a case study.” Journal of the Operational Research Society, 59, 473–481.

Borja Ponte, Enrique Sierra, David de la Fuente, Jesus Lozano (2017). “Exploring the interaction of inventory policies across the supply chain: An agent-based approach.” Computers & Operations Research Volume 78, Pages 335-348.

Bradley,E. (2003). “Intelligent Data Analysis–an Introduction.” Springer, Heidelberg, Berlin, pp. 199–227.

Brahimi, N., Dauzere-Peres, S., Najid, N. M. & Nordli, A. (2006). “Single item lot sizing problems.” European Journal of Operational Research, 168, 1‑16.

Bretthauer, K. B. Shetty, S. Syam, S. White (1994). “A model for resource constrained production and inventory management.” Decision Sciences, 25, 561–580.

Buffa, F. P., & Jackson, W. M. (1983). “A goal programming model for purchase planning.” Journal of Purchasing and Materials Management, 19, 27‑34.

Burgin, T.A., Wild, A.R. (1967). “Stock control-experience and usable theory.” Operational Research Quarterly, 18, 35‑52.

Chan, C.K., Cheung, Langevin, A. (2003). “Solving the multi-buyer joint replenishment problem with a modified genetic algorithm.” Transportation Research Part B: Methodological, 37 (3) 291‑299.

Chandra, C., Grabis, J. (2005). “Application of multi-steps forecasting for restraining the bullwhip effect and improving inventory performance under autoregressive demand”. European Journal of Operational Research, 166 (2), 337–350.

Chang, C.T., S.C. Chang (2001). “The inventory model with variable lead time and price-quantity discount.” Journal of the Operational Research Society, 52, 1151‑1158.

Chang, P., Yao, M., S. Huang, S., Chen, C. (2006). “A genetic algorithm for solving a fuzzy economic lot-size scheduling problem,” International Journal of Production Economics, 102 (2) 265‑288.

Chen, F., Drezner, Z., Ryan, J. K. and Simchi-Levi, D. (2000). “Quantifying the bullwhip effect in a simple supply chain: The impact of forecasting, lead times, and information.” Management Science, 46(3), 436‑443.

Chen-Yang Cheng, Yin-Yann Chen, Tzu-LiChen, JohnJung- WoonYoo (2015). “Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem.” Int. J.Production Economics, 17, 805–814.

Chiang, C., Gutierrez, G.J. (1996). “A periodic review inventory system with two supply modes.” European Journal of Operational Research, 94, 527–547.

Chia-Shin Chung, James Flynn, Roelof Kuik, Piotr Stalinski (2013). “A single-period inventory placement problem for a supply system with the satisficing objective.” European Journal of Operational Research, Volume 224, Issue 3, Pages 520-529.

Chi Kin Chan, Bernard, K.S., Cheung, Andre Langevin (2003). “Solving the multi-buyer joint replenishment problem with a modified genetic algorithm.” Transportation Research, Part B, 37 291–299.

Chirag Deba, Fan Zhangb, Junjing Yanga (2017). “A review on time series forecasting techniques for building energy consumption.” Renewable and Sustainable Energy Reviews, 74, 902–924.

Chiraphadhanakul, S., Dangprasert, P. & Avatchanakorn V. (1997). “Genetic algorithms in forecasting commercial banks deposits.” Proceedings of the IEEE international conference on intelligent processing systems, IEEE Press, pp. 116–210.

Chiu, M. & Lin, G. (2004). “Collaborative supply chain planning using the artificial neural network approach.” Journal of Manufacturing Technology Management, 15(8), 787–796.

Chopra, S. & Meindl, P. (2001). “Supply chain management: Strategy, planning and operation.” “Demand Management.” NJ: Prentice-Hall, Edition 5, pp. 234-270.

Christopher, M. (1992). “Logistic and Supply Chain Management.” Pitman Publishing, London, Edition 13, pp. 343-376.

Chung,C,J., Wee (2011). “Short life-cycle deteriorating product remanufacturing in a green supply chain inventory control system.” International Journal of Production Economics, 129, 195–203.

Cohen, M. A., & Ernst, R. (1988). “Multi-item classification and generic inventory stock control policies.” Production and Inventory Management Journal, 29(3), 6‑8.

Colorni, A., Dorigo, M., Maniezzo,V. (1991). “Distributed optimization by ant colonies: “Proceedings of the European Conference on Artificial Life.” Paris, France, pp. 134-142.

Company Handbook, Shalimar valves (2013). “Standard operating procedures- ABC analysis” pp. 176-190.

Cox, A., Sanderson, J., Watson, G. (2001). “Supply chains and power regimes: Towards an analytic framework for managing extended networks of buyer and supplier relationships.” Journal of Supply Chain Management, 37 (2), 28–35.

Crum, C. and Palmatier, G. E. (2003). “Demand Management Best Practices.” J. Ross Publishing, 6th Edition, pp. 435-445.

Cybenko, G. (1989). “Approximation by superposition of a sigmoidal function.” Mathematics of Control Signals and Systems, 2, 303‑314.

Das, K., Roy, Maiti, M. (2000). “Multi-item inventory model with quantity-dependent inventory costs and demand-dependent unit cost under imprecise objective and restrictions: a geometric programming approach.” Prod. Planning &. Control, 11, 781‑788.

Davood Mohammaditabar, SeyedHassan, Ghodsypour, ChrisO'Brien (2010). "Inventory control system design by integrating inventory classification and policy selection." Int. J. Production Economics, 140, 655–659.

De Carvalho, M.C.M, Dougherty, A.S. Fowkes, M.R. Wardman (1998). "Forecasting travel demand: a comparison of different artificial neural network methods." The Journal of the Operational Research Society, 49 (7), 717–722.

Dejonckheere, J., Disney, S.M., Lambrecht, M.R., Towill, D.R. (2003). "Measuring and avoiding the bullwhip effect: A control theoretic approach." European Journal of Operational Research, 147 (3), 567–590.

Devendra Choudhary, Ravi Shankar (2011). "Modeling and analysis of single item multi-period procurement lot-sizing problem considering rejections and late deliveries." Computers & Industrial Engineering, 61, 1318–1323.

Dilay Celabi (2015). "Inventory control in a centralized distribution network using genetic algorithms: A case study." Computers & Industrial Engineering, 87, 532‑539

Dorigo M and Stutzle (2004). "Ant Colony Optimization." MIT Press, Cambridge, MA, 15th edition, pp. 221-229.

Dorffner, G. (1996). "Neural Networks for Time Series Processing." Neural Network World (4), 447‑468.

Dubois, D., Prade, H. (1986). "Fuzzy sets and statistical data," Eur. J. Oper. Res. 25 (3) (1986) 345–356.

Du, T.C., Wolfe, P.M. (1997). "Implementation of fuzzy logic systems and neural networks in industry." Computers in Industry, 32, 261–272.

Dunsmuir, W.T.M., Snyder, R.D. (1989). “Control of inventories with intermittent Demand.” European Journal of Operational Research, 40 (1), 16‑21.

Escoda, I. Ortega, A., Sanz, A. & Herms, A. (1997). “Demand forecast by neuro-fuzzy techniques.” In Proceedings of the sixth IEEE international conference on fuzzy systems, pp. 1381‑1386.

Faraway, J. C., Chatfield (2008). “Time series forecasting with neural networks: a comparative study using the airline data.” Applied Statistics, 47 (2), 231–250.

Fardin Ahmadizar , Mehdi Zeynivand, Jamal Arkat (2015). “Two-level vehicle routing with cross-docking in a three-echelon supply chain: A genetic algorithm approach.” Applied Mathematical Modelling, 39, 7065–7081.

Fiestras-Janeiro, M.G., García-Jurado, Meca, Mosquera M.A. (2013). “A new cost allocation rule for inventory transportation systems.” Operations Research Letters, Volume 41, Issue 5, Pages 449-453.

Flores, B. E. & Whybark, D. C. (1987). “Implementing multiple criteria ABC analysis.” Journal of Operations Management, 7(1), 79‑84.

Frohlich, M. (2002). “Demand chain management in manufacturing and services: Web-based integration, drivers and performance.” Journal of Operations Management, 20 (6), 729‑745.

Funahashi, K. (2009). “On the approximate realization of continuous mappings by neural networks.” Neural Networks, 2, 183‑192.

G.E.P. Box, G.M. Jenkins, G. Reinsel (2004). “Time Series Analysis: Forecasting and Control”, third ed. Prentice Hall, pp: 137-158.

Garetti, M., & Taisch, M. (1999). “Neural networks in production planning and control.” Production Planning and Control, 10(4), 324‑339.

Garcia, Ibeas, Vilanova, R. (2013). “A switched control strategy for inventory control of the supply chain.” Journal of Process Control, 23, 868– 880.

George Nenes, Sofia Panagiotidou, George Tagaras (2010). “Inventory management of multiple items with irregular demand: A case study.” European Journal of Operational Research, 20 ,313‑324.

Gilbert, K. (2005). “An ARIMA supply chain model.” Management Science, 51, 305‑310.

Giachetti, Young (1997). “A parametric representation of fuzzy numbers and their arithmetic operators.” Fuzzy sets and systems, 91, 185–202.

Goh Sue-Ann, Ponnambalam, Jawahar (2012). “Evolutionary algorithms for optimal operating parameters of vendor managed inventory systems in a two-echelon supply chain.” Advances in Engineering Software, 52, 47–54.

Goldberg, D.E. (1989). “Genetic Algorithms in Search, Optimization and Machine Learning.” Addison-Wesley Publishing Company, Massachusetts, Edition 5, 105-130.

Gupta, R. K., Bhunia, A.K., Goyal, S.K. (2009). “An application of Genetic Algorithm in solving an inventory model with advance payment and interval valued inventory costs.” Mathematical and Computer Modelling, 49, 893-905.

Gurani, H. & Tang, C.S. (1999). “Optimal ordering decision with uncertain cost and demand forecast updating.” Management Science, 45(10), 1456-1462.

Guvenir, H.A., Erel (1998). “Multi criteria inventory classification using a genetic algorithm,” European Journal of Operational Research, 105 (1), 29‑37.

Hadjahmadi, A, H., Homayounpour, M. M. & Ahadi, S.M. (2008). "Robust weighted Fuzzy C means clustering." IEEE International Conference on Fuzzy Systems, 978 (1), 4244- 1819.

Hadley, G., Whiten, T.M. (1963). "Analysis of Inventory Systems," Prentice-Hall, Englewood Cliffs, NJ, Edition 3, 240-270.

Hamed Soleimani, Govindan Kannan (2015). "A hybrid particle swarm optimization and genetic algorithm for closed-loop supply chain network design in large-scale networks." Applied Mathematical Modelling, 39, 3990–4012.

Handfield, R., Warsing, D., Wu, X., (2009.) "Q, r inventory policies in a fuzzy uncertain supply chain environment." European Journal of Operational Research, 197, 609–619.

Hansen, J.V., R.D. Nelson (2003). "Forecasting and recombining time-series components by using neural networks." The Journal of the Operational Research Society, 54 (3), 307–317.

Hans-Joachim Girlich (2003). "Transaction costs in finance and inventory research." Int. J. Production Economics, 81(2), 341–350.

Harris F.W. (1913). "How many parts to make at once?" The Magazine of Management, 10, 135–152.

Heikkila, J. (2002). "From supply to demand chain management: Efficiency and customer satisfaction." Journal of Operations Management, 20 (6), 747-767.

Herbrich, R., Keilbach, M.T., Graepel, P.B.S., Obermayer, K. (2000). "Neural networks in economics: Background, applications and new developments." Computational Techniques for Modeling Learning in Economics, 11, 169‑196.

He-Yau Kanga, Amy H.I., Leeb (2010). "Inventory replenishment model using fuzzy multiple objective programming: A case study of a high-tech company in Taiwan." Applied Soft Computing, 10, 1108–1118.

Hill,T., Connor, M.O., Remus, W. (2006). "Neural networks for time series forecasts," Management Science, 42 (7), 1082‑1092.

Hindriyanto, Purnomo, Hui Weeb, Yugowati Praharsi (2012). "Two inventory review policies on supply chain configuration problem." Computers & Industrial Engineering, 63, 448–455.

Hira, D.S., Gupta, P.K. (2009). "Analysis of Inventory costs" in Operations Research, S. Chand & Company Ltd., New Delhi, India, 5th Edition, page no. 235-250.

Hobbs, B. F., Helman, U., Jitprapaikulsarn, S., Konda, S., & Maratukulam, D. (1998). "Artificial neural networks for short-term energy forecasting: Accuracy and economic value." Neuro computing, 23, 71–84.

Hsu, P. H, Wee, H.M. (2008). "Coordinated ordering decisions for products with short life cycle and variable selling price, Computers & Industrial Engineering, 54 (2008) 602–612.

Hung, J.C. (2009). "A fuzzy GARCH model applied to stock market scenario using a genetic algorithm.'' Expert Syst. Applications, 36, 11710‑11717.

Ilkay Saracoglu, Seyda Topaloglu, Timur Keskinturk (2014). "A genetic algorithm approach for multi-product multi-period continuous review inventory models." Expert Systems with Applications, Volume 41, Issue 18, Pages 8189-8202.

Jamal Shahrabi , Esmaeil Hadavandi, Shahrokh Asadi (2013). "Developing a hybrid intelligent model for forecasting problems: Case study of tourism demand time series." Knowledge-Based Systems, 43, 112‑122.

Jamshidi, H., & Jain, A. (2008). "Multi-criteria ABC inventory classification: With exponential smoothing weights." Journal of Global Business Issues, 2(1), 61

Javad Sadeghi, Seyed Taghi, Akhavan Niaki (2014). "A hybrid vendor managed inventory and redundancy allocation optimization problem in supply chain management: An NSGA-II with tuned parameters." Computers & Operations Research, 41, 53–64.

Jeong B, Jung H.S., Park N. K. (2002). "A computerized causal forecasting system using genetic algorithms in supply chain management." Journal of Systems and Software, 60, 223–37.

John Holland (1975). "Adaptation in Natural and Artificial Systems." MIT Press, Cambridge, MA. Edition 7, 172-205.

John Holland (1962)."Outline for a logical theory of adaptive systems." JACM, Vol 9, no. 3, pp. 279–314.

JiSun Shinn, Sungshin Kimn, Jang-Myung Leen (2015). "Production and inventory control of auto parts based on predicted probabilistic distribution of inventory." Digital Communications and Networks 1, 292–301.

John E. Boylan, Aris Syntetos (2012). "Forecasting in management science." Omega, Volume 40, Issue 6, Page 68-76.

Ju Y.K., Kim, C., Shim, J. C. (1997). "Genetic based fuzzy models: interest rate forecasting problems." Computers and Industrial Engineering, 33, 561–565.

Juite Wanga, Yun-Feng Shub (2004). “Fuzzy decision modeling for supply chain management.” Fuzzy Sets and Systems, 150, 107–127.

Jui-Jung Liao, Kun-Jen Chung, Kuo-Nan Huang (2013).“A deterministic inventory model for deteriorating items with two warehouses and trade credit in a supply chain system.” International Journal of Production Economics, Volume 146, Issue 2, Pages 557-565.

Jui-Tsung, Wonga, Chwen-Tzeng S., Chun-Hsien (2011). “Stochastic dynamic lot-sizing problem using bi-level programming based on artificial intelligence techniques.” Applied Mathematical Modelling, 79, 9556–9572.

Kai F, Wenhua, X. (1997). “Training neural networks with genetic algorithms for forecasting the stock price index.” In: Proceedings of the IEEE international conference on intelligent processing systems, IEEE Press, p. 401–405.

Karimi, B., Fatemi Ghomi & Wilson, J. M. (2003). “The capacitated lot sizing Problem-A review of models and algorithms.” Omega, 31(5), 365‑378.

Kartalopoulos, S.V. (1996). “Understanding Neural Networks and Fuzzy Logic.” IEEE Press, New York, Edition 9, pp: 322-350.

Katagiri, H., Ishii, H. (2002). “Fuzzy inventory problems for perishable commodities.” European Journal of Operational Research, 138, 545–553.

Khouja, M. (1999). “The single-period (news-vendor) problem: Literature review and suggestions for further research.” Omega, Int. J. Management Science, 27, 537–553.

Kim D, Kim C. (2009). “Forecasting time series with genetic fuzzy predictor ensemble.” IEEE Transactions on Fuzzy Systems, 5, 523–35.

Komgrit Leksakul, Pongsak Holimchaya chotikul, Apichat Sopadang (2015). “Forecast of off-season longan supply using fuzzy support vector regression and fuzzy artificial neural network.” Computers and Electronics in Agriculture, 118, 259‑269.

Koskivaara, E. (2004). “Neural networks in analytical review procedures.” Managerial Auditing Journal, 19(2), 191–223.

Krishna K., Chintalpundi & Moshe Kam (1998). “A noise resistant fuzzy c means algorithm for clustering.” IEEE Conference on Fuzzy systems Proceedings, 150-156.

Krone,L. (1964). “A note on economic lot sizes for multi-purpose equipment.” Management Science, 10, 461–464.

Kuo, R. J., & Xue, K. C. (1998). “An intelligent sales forecasting system through integration of artificial neural network and fuzzy neural network.” Computers in Industry, 37, 1‑15.

Kuo, R. J., Wu, P. & Wang, C. P. (2002). “An intelligent sales forecasting system through integration of artificial neural networks and fuzzy neural networks with fuzzy weight elimination” Neural Networks, 15, 909‑925.

Kuo, C. A., Reitsch, A. (1995). “Neural networks vs. conventional methods of forecasting.” Journal of Business Forecast, 14 (4), 315-327.

Kuo, R.J., Leeb, Ferani (2014). “Solving bi level linear programming problem through hybrid immune genetic and particle swarm optimization algorithm.” Applied Mathematics and Computation, 266, 1013–1026.

Lau, H.C.W., Yi Zhao (2013). “A demand forecast model using a combination of surrogate data analysis and optimal neural network approach.” Decision Support Systems, 54, 1404–1416.

Lee, H. L., Padmanabhan, V. & Whang, S. (1997). “The bullwhip effect in supply chains.” Management Review, 38, 93‑102.

Leenders, M. H., Fearon, W. England (1985). “Purchasing and Materials Management.” Richard D. Irwin, Inc., Homewood, IL, Edition 5, pp. 78-89.

Leopoldo Eduardo, Cárdenas-Barron, Gerardo Treviño Garza, Hui Ming Wee (2012). “Simple and better algorithm to solve the vendor managed inventory control.” Expert Systems with Applications, 39, 3888‑3895.

Leopoldo Eduardo, Cárdenas-Barron, Gerardo Trevino Garza (2015). “ A new approach to solve the multi-product multi-period inventory lot sizing with supplier selection problem.” Computers & Operations Research, Volume 64, Pages 225-232.

Lewis, C.D. (1982). “Industrial and Business Forecasting Methods: A Practical Guide to Exponential Smoothing and Curve Fitting.” Butterworth Scientific, London; Boston, Edition 10, pp: 435-446.

Lin Wang, Qing-Liang Fu, Yu-Rong Zeng (2012). “Continuous review inventory models with a mixture of backorders and lost sales under fuzzy demand and different decision situations.” Expert Systems with Applications, 39, 4181–4189.

Lolli, F., Gamberini, R. (2017). “Single-hidden layer neural networks for forecasting intermittent demand.” International Journal of Production Economics, Vol 183, part A, 116-128.

Longsheng Cheng, Ching-ShihTsou, Dong-YuhYang (2016). “Cost-service tradeoff analysis of reorder-point-lot-size inventory models.” Journal of Manufacturing Systems, 37, 1, 217-226.

Luis, Moncayo Martinez, David Zhang (2013). "Optimising safety stock placement and lead time in an assembly supply chain using bi-objective MAX–MIN ant system." Int. J. Production Economics, 145, 18–28.

Luis Aburto, Richard Weber (2007). "Improved supply chain management based on hybrid demand forecasts." Applied Soft Computing, 7, 136‑144.

Luxhoj, J. T. & Stensballe, B. (2006). "A hybrid econometric-neural network modeling approach for sales forecasting." International Journal of Production Economics, 43, 175‑192.

Magali R.G.Meireles & Paulo E.M. Almeida (2003). "A comprehensive review of Industrial Applicability of artificial neural networks." IEEE transactions on Industrial Electronics, Vol 50, pp. 585-601.

Mahdi Tajbakhsh, M.(2010). "On the distribution free continuous-review inventory model with a service level constraint." Computers & Industrial Engineering Volume 59, Issue 4, Pages 1022-1024.

Maiti, M. K., M. Maiti (2006). "Fuzzy inventory model with two warehouses under possibility constraints." Fuzzy Sets and Systems, 157 (1), 52‑73.

Maiti, M. K., M. Maiti (2007). "Two-storage inventory model with lot-size dependent fuzzy lead-time under possibility constraints via genetic algorithm." European Journal of Operational Research, 179 (2) 352‑371.

Maiti, A. K., M. Maiti (2008). "Discounted multi-item inventory model via genetic algorithm with roulette wheel selection, arithmetic crossover and uniform mutation in constraints bounded domains." Int. J. Computer Mathematics, 85, 1341‑1353.

Makridakis, S., Wheelwright, S. C., & Hyndman, R. J. (1998). “Forecasting: Methods and applications” (3rd ed.). New York: John Wiley & Sons, pp. 201-220.

Manuel Cardos, Eugenia Babiloni (2011). “Exact and approximated calculation of the cycle service level in a continuous review policy.” International Journal of Production Economics, Volume 133, 1, Pages 251-255.

Mark Ko, Ashutosh Tiwari, Jorn Mehnen (2002). “Soft computing in SCM” Applied Soft Computing, 10, 661‑674.

Maria Rosienkiewicz, Edward Chlebus, Jerzy Detyna (2017). “A review on time series forecasting techniques for building energy consumption.” Renewable and Sustainable Energy Reviews, 74, 902–924.

Mark Ko , Ashutosh Tiwari, Jorn Mehnen (2010). “A review of soft computing applications in supply chain management.” Applied Soft Computing ,10, 661‑674.

Maryam Akbari Kaasgari, Din Mohammad Imani, Mehdi Mahmoodjanloo (2017). “Optimizing a vendor managed inventory (VMI) supply chain for perishable products by considering discount: Two calibrated meta-heuristic algorithms.” Computers & Industrial Engineering, Volume 103, 227-241.

Matlab (2008). “Neural Network Toolbox” Mathworks, Edition 13, pp. 248-266.

Min-ChunYu (2011). “Multi-criteria ABC analysis using artificial-intelligence-based classification techniques.” Expert Systems with Applications, 38, 3416‑3421.

Minghui Lai, Weili Xue, Lindu Zhao (2016). “Cost allocation for cooperative inventory consolidation problems.” Operations Research Letters, Volume 44, Issue 6, Pages 761-765.

Mogale, Alexandre Dolgui, Rishabh Kandhway, Sri Krishna Kumar, Manoj Kumar Tiwari (2017). "A multi-period inventory transportation model for tactical planning of food grain supply chain." Computers & Industrial Engineering, Volume 110, 379-394.

Moin, N.H., Aziz A.B., (2010). "An efficient hybrid genetic algorithm for the multi-product multi-period inventory routing problem." Int. J. Production Economics, 42, 3352-3362.

Montgomery, D.C. (2000). "Design and analysis of experiments." Wiley, New York, Edition 5, 275-285.

Musilek, P., Gupta (2000). "Neural networks and fuzzy systems." in: "N.K. Sinha, M.M. Gupta" (Eds.), Soft Computing and Intelligent Systems, Academic Press, San Diego, pp: 346-367.

Najafi, A., Niaki, S., Shahsavar (2009). "A parameter-tuned genetic algorithm for the resource investment problem with discounted cash flows and generalized precedence relations." Computers & Operations Research, 36, 2994–3001.

Nikolaos Kourentzes (2013). "Intermittent demand forecasts with neural networks." Int. J. Production Economics, 14, 198–206.

Niranjan Roy & Ranjan Ganguli (2006). "Filter design using RBF neural network for improved health monitoring." Applied Soft Computing, 6(4), pp. 154- 169.

Park, J. I., Lee, Song, and Chun (2010). "TAIFEX and KOSPI 200 forecasting based on two-factor high-order fuzzy time series and particle swarm optimization." Expert Syst. Applications, 37, 959‑967.

Pal, S., Ghosh, A. (2004). "Soft computing data mining." Information Sciences, 163, 100‑320.

Partovi, F. Y., & Anandarajan, M. (2002). “Classifying inventory using artificial neural network approach.” Computers and Industrial Engineering, 41, 389‑404.

Partovi, F. Y., & Burton, J. (1993). “Using the analytic hierarchy process for ABC analysis.” International Journal of Operations &Production Management, 13(9), 29‑44.

Patel, J. N. (2011). “Accuracy Comparison of Various Techniques to Solve Machine Layout Problem.” International Journal of Advanced Research in Computer Science, *2*(1), 40-50.

Phadke, M. S. (1989). “Quality engineering using robust design.” Prentice-Hall, Upper Saddle River. Edition 8, pp. 432-445.

Phillips, D., Ravindran, Solberg, (2006). “Operations Research: Principles and Practice.” Wiley, New York, 3rd edition, pp. 240-250.

Pourakbar, M. R. Z., Farahani, Asgari (2007). “A joint economic lot-size model for an integrated supply network using genetic algorithm.” Applied Mathematics and Computation, 189 (1) 583‑596.

Prasanna Kumar, Mervin Herbert, Srikanth Rao (2013). “AI Technique Applications in Inventory Management A Review and Analysis of Literature.” International Journal of Business and Management for Tomorrow, 3, (5), pp. 105-112.

Prasanna Kumar, Mervin Herbert, Srikanth Rao (2014). “Demand forecasting using Artificial Neural Network based on different learning methods: Comparative Analysis.” International Journal for Research in Applied Science and Engineering Technology, 2 (4), pp 76-85.

Prasanna Kumar, Mervin Herbert, Srikanth Rao (2015). “Genetic algorithm approach for analysis of multi item multi period procurement lot sizing problem.” International Journal of Management (IJM),6, (12), pp. 50-58.

Pratsini, E. (2000). “The capacitated dynamic lot size problem with variable technology” Computers & Industrial Engineering, 38(4), 493‐504.

R. Roy (1990). “A Primer on the Taguchi Method.” Society of Manufacturing Engineers, New York, Edition 4, pp. 125-175.

Roy, R., Furuhashi, T., Chawdhry, P.K.(1999). “Advances in Soft Computing.” Engineering Design and Manufacturing, Springer, 40, 370-80.

Raghunathan, S. (1999). “Inter organizational collaborative forecasting and replenishment systems and supply chain implications.” Decision Sciences, 30 (4), 56-69.

Ramanathan, R. (2006). “ABC inventory classification with multiple-criteria using weighted linear optimization.” Computers and Operations Research, 33, 695‐700.

Ravi Mahendra (2010). “Industrial Statistics and Operational Management.” “Chapter 6, Forecasting technique.” Edition 4, pp. 176-185.

Real Carbonneau, Kevin Laframboise, Rustam Vahidov (2008). “Application of machine learning techniques for supply chain demand forecasting.” European Journal of Operational Research, 184, 1140–1154.

Reza Zanjirani Farahania, Mahsa Elahipanah ((2008). “A genetic algorithm to optimize the total cost and service level for just-in-time distribution in a supply chain.” Int. J. Production Economics, 111, 229–243.

Robinson, P., Narayanan, A. & Sahin, F. (2009). “Coordinated deterministic dynamic demand lot-sizing problem: A review of models and algorithms.” Omega, 37(1), 3‐15.

Rolston, D.W. (1988). “Principles of Artificial Intelligence and Expert Systems Development.” McGraw-Hill, New York, Edition 7, pp. 340-345.

Ross R. J. (1989). “Taguchi techniques for quality engineering.” McGraw-Hill, New York, Edition 4, pp: 98-112.

Ross, T. J. (2004). “Fuzzy Logic with Engineering Applications.” 2nd ed., John Wiley and Sons, West Sussex, pp. 216-225.

Ryan Chen, F., Simchi-Levi, D. (2000). “The impact of exponential smoothing forecasts on the bullwhip effect.” Naval Research Logistics, 47(4), 269‑286.

Samak-Kulkarnia S.M., Rajhansb, N.R. (2013). “Determination of Optimum Inventory Model for Minimizing Total Inventory Cost.” Procedia Engineering, 51, 803 – 809.

Sana,S.S., K.S. Chaudhuri (2008). “A deterministic EOQ model with delays in payments and price-discount offers.” Eur. J. Operations Research, 42, 509‑533.

Seyed Hamid Reza Pasandideh, Seyed Taghi Akhavan Niaki, Nafiseh Tokhmehchi B. (2011). “A parameter-tuned genetic algorithm to optimize two-echelon continuous review inventory systems.” Expert Systems with Applications, 38, 11708–11714.

Seyed Mohsen Mousavi, Vahid Hajipour, Seyed Taghi Akhavan Niaki,Najmeh Alikar (2013). “Optimizing multi-item multi-period inventory control system with discounted cash flow and inflation: Two calibrated meta-heuristic algorithms.” Applied Mathematical Modelling, 37, 2241‑2256.

Shing Chih Tsai, Sin Ting Chen (2017). “A simulation-based multi-objective optimization framework: A case study on inventory management.” Omega, Volume 70, 148-159.

Shu-Chin Chang, Ching-Ter Chang (2016). “Multi-stage and multi-supplier inventory model allowing different order quantities.” Applied Mathematical Modelling Volume 52, Pages 613-625.

Silva, C.A., J.M.C. Sousa, T. Runkler, R. Palm. (2005). "Soft computing optimization methods applied to logistic processes." International Journal of Approximate Reasoning, 40 (3), 280–301.

Silva, C.A., J.M.C. Sousa, T. Runkler, R. Palm (2005). "Soft computing optimization methods applied to logistic processes." International Journal of Approximate Reasoning, 40, 280–301.

Silver, E. A., Pyke, D., & Peterson, R. (1990). "Inventory management and production planning and scheduling (2 nd ed), John Wiley and Sons, New York, pp:157-180.

Simon Haykin (1999). "Neural Networks- A Comprehensive Foundation". Pearson Education Publishing. Edition 5, pp: 130-170.

Singh, P., Deo,M.C. (2007). "Suitability of different neural networks in daily flow forecasting." Applied Soft Computing, 7, 968–978.

Smith, N. R., Robles, J. L. & Cárdenas, L. E. (2009). "Optimal pricing and production master planning in a multi period horizon considering capacity and inventory constraints." Mathematical Problems in Engineering, 11, 1–15.

Srinivasa Pai. P., Nagabhushan, Ramakrishna Rao (2003). "Radial basis function neural networks for Tool wear monitoring." Int. Journal of COMADEM, Vol 5, pp. 521-30.

Syntetos, A. A., Boylan, J. E. (2006). "Stock control performance of intermittent demand estimators". International Journal of Production Economics 103 (1), 36– 47.

Taguchi, G., Chowdhury, S., Y. Wu (2005). "Taguchi's Quality Engineering Handbook." Wiley, New Jersey. Edition 10, pp: 232-252.

Taleizadeh, A.A., Niaki, M.B., Aryanezhad, A. & Fallah Tafti (2010). "A genetic algorithm to optimize multi-product multi-constraint inventory control systems with stochastic replenishments and discount.” Int. J. Adv. Manufacturing Technology, 51, 311‑323.

Tamas Koltai (2009). “Robustness of a production schedule to inventory cost calculations.” International Journal of Production Economics Volume 121, Issue 2, Pages 494-504.

Tan, K.C. (2001). “A framework of supply chain management literature.” European Journal of Purchasing & Supply Management, 7 (1), 39–48.

Tettamanzi, A., Tomassini, M. (2001). “Soft computing: Integrating Evolutionary, Neural and Fuzzy Systems.” Springer, Heidelberg, 45, 250-65.

Torkul, Yimalz, Selvi, Cesur, M.R. (2016). “A real-time inventory model to manage variance of demand for decreasing inventory holding cost.” Computers & Industrial Engineering, Volume 102, Pages 435-439.

Tugba Efendigil, Semih Onut, Cengiz Kahraman (2009). “A decision support system for demand forecasting with artificial neural networks and neuro-fuzzy models: A comparative analysis.” Expert Systems with Applications, 36, 6697–6707.

Vakharia, A.J. (2002). “E-business and supply chain management.” Decision Sciences, 33 (4), 495‑505.

Van Wingerden, E., Basten, R., Dekker, R., & Rustenburg, W. (2014). “ More grip on inventory control through improved forecasting: a comparative study at three companies.” International Journal of Production Economics, *157*, 220–237.

Wagner, H. M. & Whitin, T. M. (1958). “Dynamic version of the economic lot-size model”. Management Science, 5, 89‑96.

WANG Xiaobin, TANG Wansheng (2007). "Fuzzy Economic Order Quantity Inventory Models without Backordering." Tsinghua Science and Technology, 12, 91-96.

Wang, T. S. Chien (2006). "Forecasting innovation performance via neural networks – a case of Taiwanese manufacturing industry." Technovation, 26, 635– 643.

Wan Lung Ng (2007). "A simple classifier for multiple criteria ABC analysis." European Journal of Operational Research 177, 344–353.

Watson, R.B. (1987). "The effects of demand-forecast fluctuations on customer service and inventory cost when demand is lumpy." Journal of the Operational Research Society, 38 (1), 75–82.

Wei, S., Zhang, J., & Li, Z. (1997). "A supplier-selecting system using a neural network 1997". In IEEE international conference on intelligent processing systems, pp. 468– 471.

Weihui Deng, Guoyin Wang, Xuerui Zhang, JiXu GuangdiLi (2016). "A multi-granularity combined prediction model based on fuzzy trend forecasting and particle swarm techniques." Neuro computing, 173, 1671–1682.

Wong, B. K., Bodnovich, T. A. & Selvi, Y. (2007). "Neural network applications in business: A review and analysis of the literature." Decision Support Systems, 19, 301–320.

Woon Seek Leea, Jong-Han Hana, Sung-Jin Cho (2005). "A heuristic algorithm for a multi-product dynamic lot-sizing and shipping problem." Int. J. Production Economics, 98, 204–214.

Yager, R., Zadeh, L., (1994). "Fuzzy Sets, Neural Networks, and Soft Computing." Van Nostrand Reinhold Publ., New York, 50, 230-270.

Yazgi Tu , Onur Ako, Ays-en Apaydınc, Dobrila Petrovic (2008). "Continuous review inventory control in the presence of fuzzy costs." Int. J. Production Economics, 113, 775–784.

Yi Tao Loo, Hay Lee, Ek Peng Chew (2017). "Inventory control policy for a periodic review system with expediting." Applied Mathematical Modelling, 49, 375–393.

Yi Tao, Loo Hay Lee, Ek Peng Chew, Gang Sun, Vincent Charle (2017). "Inventory control policy for a periodic review system with expediting." Applied Mathematical Modelling, Volume 49, Pages 375-393.

Zadeh, L. A. (1965). "Fuzzy sets." Information and Control, 8, 338–353.

Zadeh, L.A. (1978). "Fuzzy sets as a basis for a theory of possibility," Fuzzy Sets and Systems, Volume 1, issue 3, 3–28.

Zhao, Xie, J., Wei, J.C., (2002). "The impact of forecast errors on early order commitment in a supply chain." Decision Sciences, 33 (2), 251‑280.

Zhong Yao, Ke Liu, Stephen C.H. Leung, K.K. Lai (2011). "Single period stochastic inventory problems with ordering or returns policies." Computers & Industrial Engineering, Volume 61, Issue 2, Pages 242-253.

Zoller, K.(1997). "Deterministic multi-item inventory systems with limited capacity." Management Science, 24, 451–455.

APPENDIX -I

Product range of company under study

Wedge Gate Valve

valve	rating	150	300	600	900	1500	2500
GTV 101	inches	1/2"-48"	1/2"-36"	1/2"-24"	2"-24"	1/2"-12"	1/2"-12"
	mm	15-1200	15-900	15-600	15-600	15-300	15-300

Parallel Slide Gate Valves

valve	rating	150	300	600	900	1500	2500
PGT 101	inches	2"-24"	2"-24"	2"-24"	2"-24"	2"-12"	2"-12"
	mm	50-600	50-600	50-600	50-600	50-300	50-300

Angle Globe Valve

valve	rating	150	300	600	900	1500	2500
AGV 102	inches	2"-16"	2"-16"	2"-16"	2"-10"	2"-10"	-
	mm	50-400	50-400	50-400	50-250	50-250	

Swing Check Valve

valve	rating	150	300	600	900	1500	2500
CHV 108	inches	2"-30"	2"-24"	2"-24"	2"-16"	2"-12"	2"-12"
	mm	50-750	50-600	50-600	50-400	50-300	50-300

The Wide Product Range –Industrial Valves (Source: www.shalimar valves.com)

APPENDIX II

C++ Program for Demand forecast using Neural Network with MLP architecture

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <fstream>
#include <sstream>
#include <cmath>

using namespace std;

#define INPUT 4
#define OUTLAYER 1

int Pattern_Nos,Train_Nos,Epochs,Epochs_Max;

float ****Delta,
       ***Weight_ih,
       ***Jacobbi,
        **Identity_Mue,
        **Net,
        **Slope,
        **Input_X,
        **Input_X_Test,
        **Weight_ho,
        **ActVal,
         *Output_X,
         *Desired,
         *Deter_Total,
         *D_Max,
          ERMS,
          E_Out,
          Mue,
          Beta;

class MLP_LM{

public:

MLP_LM(){
    Pattern_Nos = 66;
    Mue = 0.7;
    Beta = 0.5;
    Epochs_Max = 150;
}

void ERate_Calc(){
    int i;
    E_Out = 0.0;
    //cout<<"\nMean Square Error Calculation . . .\nPress any key to
continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++)
        E_Out += pow(Desired[i]-Output_X[i],2);
    E_Out *= 0.5;
}

void ERMS_Calc(){
    int i;
    ERMS = 0.0;
    //cout<<"\nError Rms Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++)
        ERMS += pow(Desired[i]-Output_X[i],2);
    ERMS = (float)sqrt(ERMS/Pattern_Nos);
}

void Output_Calc(){
    int i,j;
    //cout<<"\nOutput Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++){
        Output_X[i]=0.0;
        for(j=0;j<INPUT;j++)
            Output_X[i] += (ActVal[i][j]*Weight_ho[i][j]);
    }
}

void Test_Output_Calc(){
    int i,j;
    //cout<<"\nOutput Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Train_Nos;i++){
        Output_X[i]=0.0;
        for(j=0;j<INPUT;j++)
            Output_X[i] += (ActVal[i][j]*Weight_ho[i][j]);
    }
}

void Net_Calc(int I){
    int i,j;
    //cout<<"\nNet Calculation"<<I;
    //cout<<"\nPress any key to continue.";
    //_getch();
    if(I==0){
        for(j=0;j<Pattern_Nos;j++){
            Net[I][j]=0.0;
            for(i=0;i<4;i++){
                Net[I][j] += (float)(Weight_ih[j][I][i]*Input_X[j][i]);
            }
            Net[I][j] = (float)(Net[I][j] + Weight_ho[0][i]);
        }
    }
    else{
        for(j=0;j<Pattern_Nos;j++)
                Net[I][j] =
(Weight_ho[j][I]*Output_X[j]+Weight_ho[0][j]);
    }
}

void Slope_Calc(int I){
    int j,i;
    //cout<<"\nSlope Calculation"<<I;
    //cout<<"\nPress any key to continue.";
    //_getch();
    if(I==0){
        for(j=0;j<Pattern_Nos;j++){
            // sec^2x
            //float tanhx = std::tanh(Output_X[I][j]);
            //Slope[I][j]=(float)(1.0+pow(tanhx,2));
            Slope[I][j]=0.0;
            for(i=0;i<INPUT;i++)
                Slope[I][j] += Weight_ih[j][I][i];
        }
    }
    else{
        for(j=0;j<Pattern_Nos;j++)
            Slope[I][j] = Weight_ho[I][j];
    }
}

float Test_DistWeight_ih_Calc(int I,int J){
    int i;
    float Temp=0.0;
    for(i=0;i<INPUT;i++){
        Temp += pow(Weight_ih[I][J][i]-Weight_ih[I][0][i],2);
    }
    return sqrt(Temp);
}

float Test_DistInput_Calc(int I,int J){
    int i;
    float Temp=0.0;
    for(i=0;i<INPUT;i++){
        Temp += pow(Input_X_Test[I][i]-Weight_ih[I][J][i],2);
    }
    return Temp;
}

void Activation_Fcnt(int I){
    int j;
    //cout<<"\nActivation Function Calculation"<<I;
    //cout<<"\nPress any key to continue.";
    //_getch();
    this->Net_Calc(I);
    for(j=0;j<Pattern_Nos;j++)
        Net[I][j]=(float)tanh(Net[I][j]);
        //Output_X[I][j]=Net[I][j];
    this->Slope_Calc(I);
}

float DistWeight_ih_Calc(int I,int J){
    int i;
    float Temp=0.0;
    for(i=0;i<INPUT;i++){
        Temp += pow(Weight_ih[I][J][i]-Weight_ih[I][0][i],2);
    }
    return sqrt(Temp);
}

float DistInput_Calc(int I,int J){
    int i;
    float Temp=0.0;
    for(i=0;i<INPUT;i++){
        Temp += pow(Input_X[I][i]-Weight_ih[I][J][i],2);
    }
    return Temp;
}

void Identity_Mue_Calc(){
    int i,j;
    for(i=0;i<INPUT;i++)
        for(j=0;j<INPUT;j++){
            if(i==j)
                Identity_Mue[i][j]=Mue;
            else
                Identity_Mue[i][j]=0.0;
        }
}

void Test_Activation_Calc(){
    int i,j;
    //cout<<"\nActivation Function Calculation . . .\nPress any key to
continue.";
    //_getch();
    for(i=0;i<Train_Nos;i++)
        for(j=0;j<INPUT;j++)
            ActVal[i][j] = (float) exp(-INPUT*this-
>Test_DistInput_Calc(i,j)/pow(D_Max[i],2));
            //ActVal[i][j]=
(float)(1/pow((pow(DistInput_Calc(i,j),2)*pow(D_Max[i],2)),0.5));
    /*
    cout<<"\n";
    for(i=0;i<Pattern_Nos;i++){
        for(j=0;j<INPUT;j++)
            cout<<ActVal[i][j]<<" ";
        cout<<"\n";
    }
    */
```

```
}

void Test_Width_Calc(){
    int i,j;
    float Temp_Max=0.0,Temp;
    //cout<<"\n Width Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Train_Nos;i++){
        for(j=1;j<INPUT;j++){
            Temp = this->Test_DistWeight_ih_Calc(i,j);
            if(Temp>Temp_Max)
                Temp_Max = Temp;
        }
        D_Max[i]=Temp_Max;
    }
}


void Activation_Calc(){
    int i,j;
    //cout<<"\nActivation Function Calculation . . .\nPress any key to
continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++)
            ActVal[i][j] = (float) exp(-
INPUT*DistInput_Calc(i,j)/pow(D_Max[i],2));
            //ActVal[i][j]=
(float)(1/pow((pow(DistInput_Calc(i,j),2)*pow(D_Max[i],2)),0.5));
    /*
    cout<<"\n";
    for(i=0;i<Pattern_Nos;i++){
        for(j=0;j<INPUT;j++)
            cout<<ActVal[i][j]<<" ";
        cout<<"\n";
    }
    */
}

void Width_Calc(){
    int i,j;
    float Temp_Max=0.0,Temp;
    //cout<<"\n Width Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++){
        for(j=1;j<INPUT;j++){
            Temp = this->DistWeight_ih_Calc(i,j);
            if(Temp>Temp_Max)
                Temp_Max = Temp;
        }
        D_Max[i]=Temp_Max;
    }
}
```

```
float Deter_Calc(int I,float ***A,int Nor)
{
    int x,y,j1,j2,p;
    float Det;
    float ***M=NULL;
    //printf("\n\n~~~~~~~~~~DETER CALCULATION~~~~~~~~~~\n\n");
    if(Nor == 1.0){
        Det = A[I][0][0];
    }
    else if (Nor == 2.0){
        Det = A[I][0][0] * A[I][1][1] - A[I][1][0] * A[I][0][1];
    }
    else {
        Det = 0.0;
        for(j1=0;j1<Nor;j1++){
            M = new float ** [Pattern_Nos];
            for(p=0;p<Pattern_Nos;p++){
                M[p]= new float * [Nor-1];
                for (x=0;x<Nor-1;x++)
                    M[p][x] = new float [Nor-1];
            }
            for (x=1;x<Nor;x++){
                j2 = 0;
                for (y=0;y<Nor;y++){
                    if (y == j1)
                        continue;
                    M[I][x-1][j2] = A[I][x][y];
                    j2++;
                }
            }
            Det += pow(-1.0,j1+2.0) * A[I][0][j1] * this-
>Deter_Calc(I,M,Nor-1);
            for(p=0;p<Pattern_Nos;p++){
                for (x=0;x<Nor-1;x++)
                    delete(M[p][x]);
                delete(M[p]);
            }
            delete(M);
        }
    }
    return(Det);
}

float ** CoFactors_Calc(int I,float ***A,int Nor)
{
    int x,y,ii,jj,i1,j1,p;
    float Det;
    float ***C,**B;
    //printf("\n\n~~~~~~~~~~~~COFACTOR MATRIX~~~~~~~~~~~~\n\n");
    B = new float * [Nor];
    for (x=0;x<Nor;x++)
        B[x] = new float [Nor];
    C = new float ** [Pattern_Nos];
    for(p=0;p<Pattern_Nos;p++){
```

```
        C[p]= new float * [Nor-1];
        for (x=0;x<Nor-1;x++)
            C[p][x] = new float [Nor-1];
    }
    for (y=0;y<Nor;y++){
        for (x=0;x<Nor;x++){
            i1 = 0;
            for (ii=0;ii<Nor;ii++){
                if (ii == x)
                    continue;
                j1 = 0;
                for (jj=0;jj<Nor;jj++){
                    if (jj == y)
                        continue;
                    C[I][i1][j1] = A[I][ii][jj];
                    j1++;
                }
                i1++;
            }
            Det = this->Deter_Calc(I,C,Nor-1);
            B[x][y] = pow(-1.0,x+y+2.0) * Det;
        }
    }
    for(p=0;p<Pattern_Nos;p++){
        for(x=0;x<Nor-1;x++)
            delete(C[p][x]);
        delete(C[p]);
    }
    delete(C);

   return B;
}

float** MatTrans_Calc(int I,float *** Mat){
    int i,j,m;
    //cout<<"\nTrans Matrix Calculation";
    //cout<<"\nPress any key to continue.";
    //_getch();
    float ** MatTrans;
    MatTrans = new float * [INPUT];
    for(i=0;i<INPUT;i++)
        MatTrans [i] = new float [INPUT];
    for(m=0;m<INPUT;m++)
        for(j=0;j<INPUT;j++)
            MatTrans[m][j] = Mat[I][j][m];
    return MatTrans;
}

float *** Inverse_Calc(float *** Num){
    //cout<<"\nInverse Calculation";
    //cout<<"\nPress any key to continue.";
    //_getch();
    int i,x, y;
    float *** Fac, *** Inv, *Dt;
    Dt = new float [Pattern_Nos];
    Inv = new float **[Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++){
        Inv[i] = new float * [INPUT];
        for(x=0;x<INPUT;x++)
            Inv [i][x] = new float [INPUT];
    }
    Fac = new float **[Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++){
        Fac [i]= new float * [INPUT];
        for(x=0;x<INPUT;x++)
            Fac [i][x] = new float [INPUT];
    }
    for(i=0;i<Pattern_Nos;i++){
        Fac[i] = this->CoFactors_Calc(i,Num,INPUT);
        Inv[i] = this->MatTrans_Calc(i,Fac);
        Dt[i] = this->Deter_Calc(i,Num,INPUT);
        //cout<<"To Deter for inverse";
        for(x=0;x<INPUT;x++) {
            for(y=0;y<INPUT;y++) {
                Inv[i][x][y] = (float)Inv[i][x][y]/Dt[i];
            }
        }
    }
    return (Inv);
}

float *** MatMultiply_Calc(float *** a, float *** b){
    int i,j,m,k;
    //cout<<"\nMultiple Matrix Calculation";
    //cout<<"\nPress any key to continue.";
    //_getch();
    float ***c;
    c = new float ** [Pattern_Nos];
    for(k=0;k<Pattern_Nos;k++){
        c[k]= new float * [INPUT];
        for(i=0;i<INPUT;i++)
            c[k][i] = new float [INPUT];
    }
    for(k=0;k<Pattern_Nos;k++)
        for(i=0;i<INPUT;i++){
            for(j=0;j<INPUT;j++){
                c[k][i][j]=0;
                for(m=0;m<INPUT;m++)
                    c[k][i][j]=c[k][i][j]+a[k][i][m]*b[k][m][j];
            }
        }
    return c;
}

float *** MatAdd_Calc(float *** a, float ** b){
    int i,j,k;
    //cout<<"\nAdd Matrix Calculation";
    //cout<<"\nPress any key to continue.";
    //_getch();
    float ***c;
    c = new float **[Pattern_Nos];
    for(k=0;k<Pattern_Nos;k++){
        c[k] = new float * [INPUT];
        for(i=0;i<INPUT;i++)
            c[k][i] = new float [INPUT];
    }
    for(k=0;k<Pattern_Nos;k++)
        for(i=0;i<INPUT;i++)
            for(j=0;j<INPUT;j++)
                c[k][i][j]=a[k][i][j]+b[i][j];
    return c;
}


void Instance_Calc(){
    int k,i;
    //cout<<"\nInstance Calculation";
    //cout<<"\nPress any key to continue.";
    //_getch();
    float ***Ans1, ***Ans2;
    Ans1 = new float ** [Pattern_Nos];
    for(k=0;k<Pattern_Nos;k++){
        Ans1[k] = new float * [INPUT];
        for(i=0;i<INPUT;i++)
            Ans1[k][i] = new float [INPUT];
    }
    Ans2 = new float ** [Pattern_Nos];
    for(k=0;k<Pattern_Nos;k++){
        Ans2[k] = new float * [INPUT];
        for(i=0;i<INPUT;i++)
            Ans2[k][i] = new float [INPUT];
    }
    for(i=0;i<Pattern_Nos;i++)
        Ans2[i]=this->MatTrans_Calc(i,Jacobbi);
    Ans1=this->MatMultiply_Calc(Ans2,Jacobbi);
    this->Identity_Mue_Calc();
    Ans1=this->MatAdd_Calc(Ans1,Identity_Mue);
    Ans1=this->Inverse_Calc(Ans1);
    Ans2=this->MatMultiply_Calc(Ans1,Jacobbi);
    //cout<<"\nHellllllllldffffffffff\n";
    for(i=0;i<Pattern_Nos;i++)
        Deter_Total[i]=this->Deter_Calc(i,Ans2,INPUT);
    //cout<<"\nHellllllllldffffffffff\n";
}

void JacobbiMat_Calc(){
    int i,j,m;
    //cout<<"\nJacobbi Matrix Calculation";
    //cout<<"\nPress any key to continue.";
    //_getch();

    for(i=0;i<Pattern_Nos;i++)
```

```
        for(m=0;m<INPUT;m++)
            for(j=0;j<INPUT;j++)
                Jacobbi[i][m][j] = (float)( - Delta[0][i][m][j]*
Input_X[i][m] );
}

void Delta_JJ(int I){
    int i,j;
    //cout<<"\nDelta_JJ Calculation"<<I;
    //cout<<"\nPress any key to continue.";
    //_getch();
        for(i=0;i<Pattern_Nos;i++)
            for(j=0;j<INPUT;j++)
                Delta[I][i][j][j] = Slope[I][i];
}

void Delta_JK(int I){
    int i,j,m;
    //cout<<"\nDelta_JK Calculation"<<I;
    //cout<<"\nPress any key to continue.";
    //_getch();
    if(I==OUTLAYER){
        this->Delta_JJ(I);
        for(i=0;i<Pattern_Nos;i++)
            for(j=0;j<INPUT;j++)
                for(m=0;m<INPUT;m++)
                    if(j!=m)
                        Delta[I][i][j][m]=0;
    }
    else{
        for(i=0;i<Pattern_Nos;i++)
            for(j=0;j<INPUT;j++)
                for(m=0;m<INPUT;m++)
                    Delta[I][i][j][m] =
(float)(Weight_ho[I][i]*Delta[1][i][j][j]);
        for(i=0;i<Pattern_Nos;i++)
            for(j=0;j<INPUT;j++)
                for(m=0;m<INPUT;m++)
                    Delta[I][i][j][m] =
(float)(Delta[I][i][j][m]*Slope[I][i]);
        this->JacobbiMat_Calc();
    }
}

void ForBack_Pass(){
    int fdw;
    //cout<<"\nForward Pass Calculation";
    //cout<<"\nPress any key to continue.";
    //_getch();
    for(fdw=0;fdw<2;fdw++)
        this->Activation_Fcnt(fdw);
    for(fdw=1;fdw>=0;fdw--)
        this->Delta_JK(fdw);
}
```

```
void Update_Weight(){
    int i,j;
    //cout<<"\nUpdate Weight . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++)
            Weight_ho[i][j] += (Beta*(Desired[i]-
Output_X[i])*ActVal[i][j]);
}

void Update_Center(){
    int i,j,k;
    //cout<<"\nUpdate Center . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++)
            for(k=0;k<INPUT;k++)
                Weight_ih[i][j][k] += (2*INPUT*Mue*(Desired[i]-
Output_X[i])*Weight_ho[i][j]*ActVal[i][j]*(Input_X[i][j]-
Weight_ih[i][j][k])/pow(D_Max[i],2));
}

float GetRand(float Max){
    return ((-Max-Max)*((float)rand()/RAND_MAX))+Max;
}

void Initiate_Weight(){
    int i,j;
    //cout<<"\nInitiate Weight . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<INPUT;i++)
        Weight_ho[0][i] = GetRand(0.5);
    for(i=1;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++)
            Weight_ho[i][j] = Weight_ho[0][j];
    /*
    cout<<"\n";
    for(i=0;i<Pattern_Nos;i++){
        for(j=0;j<INPUT;j++)
            cout<<Weight_ho[i][j]<<" ";
        cout<<"\n";
    }
    */
}

void Weight_ih_Selection(){
    float Min=1000.0,Max=0.0,Temp;
    int i,j,k,I_Min,I_Max;
    time_t timev;
    //cout<<"\nCenter Selection . . .";
    //cout<<"\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++){
        Temp = 0.0;
        for(j=0;j<INPUT;j++)
            Temp += Input_X[i][j];
        if(Temp>Max){
            Max = Temp;
            I_Max = i;
        }
        if(Temp<Min){
            Min = Temp;
            I_Min = i;
        }
    }
    //cout<<"\nMin = "<<I_Min<<"\tMax = "<<I_Max;
    for(j=0;j<INPUT;j++){
        Weight_ih[0][0][j] = Input_X[I_Min][j];
        Weight_ih[0][INPUT-1][j] = Input_X[I_Max][j];
    }
    srand((unsigned) time(&timev));
    k = rand()%Pattern_Nos;
    for(j=1;j<INPUT-1;j++){
        for(i=0;i<INPUT;i++)
            Weight_ih[0][j][i] = Input_X[k][i];
        k = (k+97)%Pattern_Nos;
    }
    for(i=1;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++)
            for(k=0;k<INPUT;k++)
                Weight_ih[i][j][k] = Weight_ih[0][j][k];
    /*
    for(i=0;i<Pattern_Nos;i++){
        for(j=0;j<INPUT;j++){
            for(k=0;k<INPUT;k++)
                cout<<Weight_ih[i][j][k]<<" ";
            cout<<"    ";
        }
        cout<<"\n";
    }
    */
}

void Initiate_Data(){
    float Ip;
    int i=0,j=0,set;
    //cout<<"\nInitiating Data . . .";
    //cout<<"\nPress any key to continue.";
    //_getch();
    std::ifstream MyCsvFile("Train_Data.csv");
    if(!MyCsvFile)
        cout<<"Cant open file";
    std::string Line;
    while(getline(MyCsvFile,Line)){
        i=0;
        std::stringstream Line_stream(Line);
        std::string value;
        while(getline(Line_stream,value,',')){
            set=0;
            //cout<<"Record "<<record<<"\n";
            Ip=::atof(value.c_str());
            if(Ip){
                set=1;
                if(i==4){
                    Desired[j]=Ip;
                    //cout<<Desired[j];
                }
                else{
                    Input_X[j][i]=Ip;
                    //cout<<Neuron[j][i];
                }
            i++;
            }
            //cout<<"\n";
        }
        if(set)
            j++;
    }
}

void Test_Initiate_Data(){
    int i,j,m=0,p=1,k;
    for(i=0;i<Pattern_Nos;i++){
        if(p%5==0){
            for(j=0;j<INPUT;j++)
                Input_X_Test[m][j] = Input_X[i][j];
            m++;
        }
        p++;
    }
    for(i=1;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++){
            Weight_ho[0][j] += Weight_ho[i][j];
            for(k=0;k<INPUT;k++)
                Weight_ih[0][j][k] += Weight_ih[i][j][k];
        }
    for(i=0;i<INPUT;i++){
        Weight_ho[0][i] /= Pattern_Nos;
        for(j=0;j<INPUT;j++)
            Weight_ih[0][i][j] /= Pattern_Nos;
    }
    for(i=1;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++){
            Weight_ho[i][j] = Weight_ho[0][j];
            for(k=0;k<INPUT;k++)
                Weight_ih[i][j][k] = Weight_ih[0][j][k];
        }
}

void InitiateInput_Data(){
    float Ip;
    int i=0,j=0,set;
    //cout<<"\nInitiating Data . . .";
    //cout<<"\nPress any key to continue.";
    //_getch();
    std::ifstream MyCsvFile("Input.csv");
    if(!MyCsvFile)
        cout<<"Cant open file";
    std::string Line;
    while(getline(MyCsvFile,Line)){
        i=0;
        std::stringstream Line_stream(Line);
        std::string value;
        while(getline(Line_stream,value,',')){
            set=0;
            //cout<<"Record "<<record<<"\n";
            Ip=::atof(value.c_str());
            if(Ip){
                set=1;
                if(i==4){
                    Desired[j]=Ip;
                    //cout<<Desired[j];
                }
                else{
                    Input_X[j][i]=Ip;
                    //cout<<Neuron[j][i];
                }
            i++;
            }
            //cout<<"\n";
        }
        if(set)
            j++;
    }
}


void Print_Learn(){
    int i;
    std::ofstream Myfile;
    Myfile.open("Train_output.csv");
    Myfile<<",Inputs,,,,Hidden-Weights\n";
    for(i=0;i<Pattern_Nos;i++)

Myfile<<Input_X[i][0]<<","<<Input_X[i][1]<<","<<Input_X[i][2]<<","<<Inpu
t_X[i][3]<<","<<Weight_ho[i][0]<<","<<Weight_ho[i][1]<<","<<Weight_ho[i]
[2]<<","<<Weight_ho[i][3]<<"\n";
    Myfile.close();
}

void Print_Test(){
    int i;
    std::ofstream Myfile;
    Myfile.open("Test_output.csv");
    Myfile<<",Inputs,,,Outputs\n";
    for(i=0;i<Train_Nos;i++)

Myfile<<Input_X_Test[i][0]<<","<<Input_X_Test[i][1]<<","<<Input_X_Test[i
][2]<<","<<Input_X_Test[i][3]<<","<<Output_X[i]<<"\n";
    Myfile.close();
}

void Print_Output(){
    int i;
    std::ofstream Myfile;
    Myfile.open("Output.csv");
    Myfile<<",Inputs,,,Outputs\n";
    for(i=0;i<Pattern_Nos;i++)

Myfile<<Input_X[i][0]<<","<<Input_X[i][1]<<","<<Input_X[i][2]<<","<<Inpu
t_X[i][3]<<","<<Output_X[i]<<"\n";
    Myfile.close();
}


void Allocate_Memory(){
    int i,j,k;
    char ch;

    cout<<"\nNumber of Patterns = "<<Pattern_Nos<<"\nDo you want to
change (y/n)";
    cin>>ch;
    if(ch=='y'||ch=='Y'){
        cout<<"Enter Number of patterns : ";
        cin>>Pattern_Nos;
    }
    Train_Nos = Pattern_Nos/5;
    cout<<"\nEpochs Limit = "<<Epochs_Max<<"\nDo you want to change
(y/n)";
    cin>>ch;
    if(ch=='y'||ch=='Y'){
        cout<<"Enter the Epochs Limit : ";
        cin>>Epochs_Max;
    }
    //cout<<"\nAllocating Memory\nPress any key to continue.";
    //_getch();
    Weight_ih = new float ** [Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++){
        Weight_ih [i] = new float * [INPUT];
        for(j=0;j<INPUT;j++)
            Weight_ih [i][j] = new float [INPUT];
    }
    Input_X = new float * [Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++)
        Input_X[i] = new float [INPUT];
    Input_X_Test = new float * [Train_Nos];
    for(i=0;i<Train_Nos;i++)
        Input_X_Test[i] = new float [INPUT];
    Weight_ho = new float * [Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++)
        Weight_ho [i] = new float [INPUT];
    ActVal = new float * [Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++)
        ActVal [i] = new float [INPUT];
    Output_X = new float [Pattern_Nos];
    Desired = new float [Pattern_Nos];
    D_Max = new float [Pattern_Nos];
    Net = new float * [2];
    for(i=0;i<2;i++)
        Net [i] = new float [Pattern_Nos];
    Slope = new float * [2];
    for(i=0;i<2;i++)
        Slope [i] = new float [Pattern_Nos];
    //cout<<"\nAllocating Memory . . .";
    Delta = new float *** [2];
    for(i=0;i<2;i++){
        Delta[i] = new float ** [Pattern_Nos];
        for(j=0;j<Pattern_Nos;j++){
            Delta [i][j] = new float * [INPUT];
            for(k=0;k<INPUT;k++)
                Delta [i][j][k] = new float [INPUT];
        }
    }
    Deter_Total = new float [Pattern_Nos];
    //cout<<"\nAllocating Memory . . .";
    Jacobbi = new float ** [Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++){
        Jacobbi[i] = new float * [INPUT];
        for(j=0;j<INPUT;j++)
            Jacobbi[i][j] = new float [INPUT];
    }
    Identity_Mue = new float * [INPUT];
    for(i=0;i<INPUT;i++)
        Identity_Mue[i] = new float [INPUT];
}

void Real_Phase(){
    //this->InitiateInput_Data();
    this->Width_Calc();
    this->Activation_Calc();
    this->Output_Calc();
    this->Print_Output();
    cout<<"\nExecution Phase executed successfully\n";
}

void Testing_Phase(){
    //int i;
    this->Test_Initiate_Data();
    this->Test_Width_Calc();
    this->Test_Activation_Calc();
    this->Test_Output_Calc();
    this->Print_Test();
    cout<<"\nTesting Phase executed successfully\n";
    /*
    cout<<"\n~~~~~~~~~~~~~~~~~~TEST OUTPUT~~~~~~~~~~~~~~~~~~~~~~~~\n";
    for(i=0;i<Train_Nos;i++)
        cout<<Input_X[i][0]<<"  "<<Input_X[i][1]<<"  "<<Input_X[i][2]<<"
"<<Input_X[i][3]<<"    "<<Output_X[i]<<"\n";
    */
}

void Initiate_phase(){
    this->Width_Calc();
    this->ForBack_Pass();
    this->Instance_Calc();
    this->Activation_Calc();
    this->Output_Calc();
    this->ERate_Calc();
    this->ERMS_Calc();
    cout<<"\nEpochs = "<<Epochs<<"\tMean Square Error =
"<<E_Out<<"\tError RMS Value = "<<ERMS;
    Epochs++;
}

void Learning_Phase(){
    //int i;
    Epochs = 1;
    while(Epochs<=Epochs_Max){
        this->Initiate_phase();
        this->Update_Center();
        this->Update_Weight();
        //cout<<"\nEpochs = "<<Epochs<<"\tMean Square Error =
"<<E_Out<<"\tError RMS Value = "<<ERMS;
    }
    /*
    cout<<"\n\n~~~~~~~~~~~~~~~~~~~~learning Weight~~~~~~~~~~~~~~~~~\n";
    for(i=0;i<Pattern_Nos;i++)
        cout<<Weight[i][0]<<"  "<<Weight[i][1]<<"  "<<Weight[i][2]<<"
"<<Weight[i][3]<<"\n";
    */
    cout<<"\n\nLearning Phase executed successfully\n";
}

void Configuring_Phase(){
    this->Allocate_Memory();
    this->Initiate_Data();
    this->Weight_ih_Selection();
    this->Initiate_Weight();
}
};

int main(){
    int i;
    MLP_LM obj;
    cout<<"\n\n~~~~~~~~~~WELCOME TO MULTILAYER PERCEPTRON NETWORK -
RANDOM CENTER~~~~~~~~~~~~~\n\n";
```

```
    while(1){
        cout<<"\n\nKindly choose the option below\n\n    1.Configuration
\n    2.Learning Phase \n    3.Testing Phase \n    4.Execution \n
5.Exit\n";
        cin>>i;
        switch(i){
            case 1: {
                    obj.Configuring_Phase();
                    break;
            }
            case 2: {
                    obj.Learning_Phase();
                    break;
            }
            case 3:{
                    obj.Testing_Phase();
                    break;
            }
            case 4:{
                    obj.Real_Phase();
                    break;
            }
            default:exit(0);
        }
    }
        /*
        cout<<"\n";
        for(i=0;i<Pattern_Nos;i++)
        cout<<Input_X[i][0]<<" "<<Input_X[i][1]<<" "<<Input_X[i][2]<<"
"<<Input_X[i][3]<<" "<<Desired[i]<<"\n";
        */

    return 0;
}
```

C++ Program for Demand forecast using Neural Network with RBF architecture

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <fstream>
#include <sstream>
#include <cmath>

using namespace std;

#define INPUT 4

int Pattern_Nos,Train_Nos,Epochs,Epochs_Max,C_Nos;

float ***Center,
       **Input_X,
       **Input_X_Test,
       **Weight,
       **PhiVal,
       **U_Part,
       **U_Part1,
       **A_Mat,
        *V_Center,
        *Output_X,
        *Desired,
        *D_Max,
         M_Exp,
         E_Toler,
         E_Toler_Out,
         ERMS,
         E_Out,
         Nue1,
         Nue2;


class RBF_FCM{

public:

RBF_FCM(){
    Pattern_Nos = 66;
    Nue1 = 0.5;
    Nue2 = 0.3;
    Epochs_Max = 40;
    M_Exp = 2.0;
    E_Toler = 0.01;
}

void ERate_Calc(){
    int i;
    E_Out = 0.0;
    //cout<<"\nMean Square Error Calculation . . .\nPress any key to
continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++)
        E_Out += pow(Desired[i]-Output_X[i],2);
    E_Out *= 0.5;
}

void ERMS_Calc(){
    int i;
    ERMS = 0.0;
    //cout<<"\nError Rms Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++)
        ERMS += pow(Desired[i]-Output_X[i],2);
    ERMS = (float)sqrt(ERMS/Pattern_Nos);
}

void Output_Calc(){
    int i,j;
    //cout<<"\nOutput Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++){
        Output_X[i]=0.0;
        for(j=0;j<INPUT;j++)
            Output_X[i] += (PhiVal[i][j]*Weight[i][j]);
    }
}

void Test_Output_Calc(){
    int i,j;
    //cout<<"\nOutput Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Train_Nos;i++){
        Output_X[i]=0.0;
        for(j=0;j<INPUT;j++)
            Output_X[i] += (PhiVal[i][j]*Weight[i][j]);
    }
}

float Test_DistCenter_Calc(int I,int J){
    int i;
    float Temp=0.0;
    for(i=0;i<INPUT;i++){
        Temp += pow(Center[I][J][i]-Center[I][0][i],2);
    }
    return sqrt(Temp);
}

float Test_DistInput_Calc(int I,int J){
    int i;
    float Temp=0.0;
    for(i=0;i<INPUT;i++)
      Temp += pow(Input_X_Test[I][i]-Center[I][J][i],2);
    return Temp;
}

void Test_Activation_Calc(){
    int i,j;
    //cout<<"\nActivation Function Calculation . . .\nPress any key to
continue.";
    //_getch();
    for(i=0;i<Train_Nos;i++)
        for(j=0;j<INPUT;j++){
            //cout<<this->Test_DistInput_Calc(i,j)<<"  ";
            PhiVal[i][j] = (float) exp(-INPUT*this-
>Test_DistInput_Calc(i,j)/pow(D_Max[i],2));
        }
    /*
    cout<<"\n";
    for(i=0;i<Train_Nos;i++){
        for(j=0;j<INPUT;j++)
            cout<<PhiVal[i][j]<<" ";
        cout<<"\n";
    }
    */
}

void Test_Width_Calc(){
    int i,j;
    float Temp_Max=0.0,Temp;
    //cout<<"\n Width Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Train_Nos;i++){
        for(j=1;j<INPUT;j++){
            Temp = this->Test_DistCenter_Calc(i,j);
            if(Temp>Temp_Max)
                Temp_Max = Temp;
        }
        D_Max[i]=Temp_Max;
    }
    /*
    for(i=0;i<Train_Nos;i++)
        cout<<D_Max[i]<<"  ";
    */
}


float DistCenter_Calc(int I,int J){
    int i;
    float Temp=0.0;
    for(i=0;i<INPUT;i++){
        Temp += pow(Center[I][J][i]-Center[I][0][i],2);
    }
    return sqrt(Temp);
}

float DistInput_Calc(int I,int J){
    int i;
    float Temp=0.0;
    for(i=0;i<INPUT;i++)
      Temp += pow(Input_X[I][i]-Center[I][J][i],2);
    return Temp;
}

void Activation_Calc(){
    int i,j;
    //cout<<"\nActivation Function Calculation . . .\nPress any key to
continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++){
        //cout<<D_Max[i]<<" ";
        for(j=0;j<INPUT;j++){
            //cout<<this->DistInput_Calc(i,j)<<" ";
            PhiVal[i][j] = (float) exp(-INPUT*this-
>DistInput_Calc(i,j)/pow(D_Max[i],2));
        }
    }
    /*
    cout<<"\n";
    for(i=0;i<Pattern_Nos;i++){
        for(j=0;j<INPUT;j++)
            cout<<PhiVal[i][j]<<" ";
        cout<<"\n";
    }
    */
}

void Width_Calc(){
    int i,j;
    float Temp_Max=0.0,Temp;
    //cout<<"\n Width Calculation . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++){
        for(j=1;j<INPUT;j++){
            Temp = this->DistCenter_Calc(i,j);
            if(Temp>Temp_Max)
                Temp_Max = Temp;
        }
        D_Max[i]=Temp_Max;
    }
    /*
    for(i=0;i<Train_Nos;i++)
```

```
        cout<<D_Max[i]<<"  ";
    */
}

void Update_Weight(){
    int i,j;
    //cout<<"\nUpdate Weight . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++)
            Weight[i][j] += (Nue2*(Desired[i]-
Output_X[i])*PhiVal[i][j]);
}

void Update_Center(){
    int i,j,k;
    //cout<<"\nUpdate Center . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++)
            for(k=0;k<INPUT;k++)
                Center[i][j][k] += (2*INPUT*Nue1*(Desired[i]-
Output_X[i])*Weight[i][j]*PhiVal[i][j]*(Input_X[i][j]-
Center[i][j][k])/pow(D_Max[i],2));
}

float GetRand(float Max){
    return ((-Max-Max)*((float)rand()/RAND_MAX))+Max;
}

void Initiate_Weight(){
    int i,j;
    //cout<<"\nInitiate Weight . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<INPUT;i++)
        Weight[0][i] = GetRand(0.5);
    for(i=1;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++)
            Weight[i][j] = Weight[0][j];
    /*
    cout<<"\n";
    for(i=0;i<Pattern_Nos;i++){
        for(j=0;j<INPUT;j++)
            cout<<Weight[i][j]<<" ";
        cout<<"\n";
    }
    */
}

float U_Mue_Input_Calc(int I){
    int i;
    float MueZ=0.0;
    //cout<<"\nU_Mue_Input_Calc Data . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<INPUT;i++)
        MueZ += (float)(pow(U_Part[I][i],M_Exp)*Input_X[I][i]);
    return MueZ;
}

float U_Mue_Calc(int I){
    int i;
    float Mue=0.0;
    //cout<<"\nU_Mue_Calc Data . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<INPUT;i++)
        Mue += pow(U_Part[I][i],M_Exp);
    return Mue;
}

void V_Center_Calc(){
    int i;
    //cout<<"\nV_Calc Data . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<C_Nos;i++)
        V_Center[i]=(float)(this->U_Mue_Input_Calc(i)/this-
>U_Mue_Calc(i));
}

float Dist_Input_V_Calc(int I,int K){
    int i;
    float Dist = 0.0;
    for(i=1;i<C_Nos;i++)
        Dist += pow(Input_X[i][K],2);
    Dist += pow((Input_X[0][K]-V_Center[I]),2);
    return sqrt(Dist);
}

float Dist_A_V_Calc(int I,int K){
    int i;
    float Dist=0.0;
    for(i=0;i<C_Nos;i++)
        Dist += pow(A_Mat[i][K],2);
    Dist += pow(A_Mat[0][K]-V_Center[I],2);
    return sqrt(Dist);
}

void A_Mat_Calc(){
    int i,k;
    //cout<<"\nA_Mat Calc Data . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<C_Nos;i++)
        for(k=0;k<INPUT;k++)
            A_Mat[i][k] = sqrt(this->Dist_Input_V_Calc(i,k)*this-
>Dist_A_V_Calc(i,k));
}

float Differ_U_Part_Calc(int K){
    int i;
    float Part = 0.0;
    for(i=0;i<C_Nos;i++)
        if(i!=K)
            Part += U_Part1[i][K];
    return Part;
}

float Div_A_Mat_Calc(int I,int K){
    int i;
    float Div=0.0,power=2/(M_Exp-1);
    for(i=0;i<C_Nos;i++)
        Div += pow((float)(A_Mat[I][K]/A_Mat[i][K]),power);
    return Div;
}

void U_Part_Calc(){
    int i,k;
    //cout<<"\nU_Part Data . . .\nPress any key to continue.";
    //_getch();
    for(i=0;i<C_Nos;i++)
        for(k=0;k<INPUT;k++){
            if(A_Mat[i][k]>0.0)
                U_Part1[i][k] = 1/this->Div_A_Mat_Calc(i,k);
            else if(A_Mat[i][k]==0.0)
                U_Part1[i][k] = 0.0;
        }
    for(i=0;i<C_Nos;i++)
        for(k=0;k<INPUT;k++){
            if(A_Mat[i][k]<0.0)
                U_Part1[i][k] = 1-this->Differ_U_Part_Calc(k);
        }
}

void E_Toler_Calc(){
    int i,k;
    //cout<<"\nE_Toler Calc Data . . .\nPress any key to continue.";
    //_getch();
    float E_M=0.0;
    for(i=0;i<C_Nos;i++)
        for(k=0;k<INPUT;k++)
            if(abs(U_Part1[i][k]-U_Part[i][k])>E_M)
                E_M = abs(U_Part1[i][k]-U_Part[i][k]);
    E_Toler_Out = E_M;
}

void Fuzzy_C_Means(){
    do{
        this->V_Center_Calc();
        this->A_Mat_Calc();
        this->U_Part_Calc();
        this->E_Toler_Calc();
    }
    while(E_Toler_Out<E_Toler);
}
```

```
void Center_Selection(){
    int i,j,k,n=0;
    //cout<<"\nCenter Selection . . .";
    //cout<<"\nPress any key to continue.";
    //_getch();
    this->Fuzzy_C_Means();
    /*
    cout<<"\n~~~~~~~~~~~~~~Fuzzy Center~~~~~~~~~~~~~~~\n\n";
    for(k=0;k<C_Nos;k++)
        cout<<V_Center[k]<<"  ";
    */
    for(i=0;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++)
              for(k=0;k<INPUT;k++){
                     Center[i][j][k] = V_Center[n];
                     n++;
                     n %= Pattern_Nos;
              }
    /*
    for(i=0;i<Pattern_Nos;i++){
        for(j=0;j<INPUT;j++){
            for(k=0;k<INPUT;k++)
                cout<<Center[i][j][k]<<" ";
            cout<<"    ";
        }
        cout<<"\n";
    }
    */
}

void Initiate_Data(){
    float Ip;
    int i=0,j=0,set;
    //cout<<"\nInitiating Data . . .";
    //cout<<"\nPress any key to continue.";
    //_getch();
    std::ifstream MyCsvFile("Train_Data.csv");
    if(!MyCsvFile)
        cout<<"Cant open file";
    std::string Line;
    while(getline(MyCsvFile,Line)){
        i=0;
        std::stringstream Line_stream(Line);
        std::string value;
        while(getline(Line_stream,value,',')){
            set=0;
            //cout<<"Record "<<record<<"\n";
            Ip=::atof(value.c_str());
            if(Ip){
                set=1;
                if(i==4){
                    Desired[j]=Ip;
                    //cout<<Desired[j];
```

```
                }
                else{
                    Input_X[j][i]=Ip;
                    //cout<<Neuron[j][i];
                }
            i++;
            }
            //cout<<"\n";
        }
        if(set)
            j++;
    }
}

void Test_Initiate_Data(){
    int i,j,m=0,p=1,k;
    for(i=0;i<Pattern_Nos;i++){
        if(p%5==0){
            for(j=0;j<INPUT;j++)
                Input_X_Test[m][j] = Input_X[i][j];
            m++;
        }
        p++;
    }
    for(i=1;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++){
            Weight[0][j] += Weight[i][j];
            for(k=0;k<INPUT;k++)
                Center[0][j][k] += Center[i][j][k];
        }
    for(i=0;i<INPUT;i++){
        Weight[0][i] /= Pattern_Nos;
        for(j=0;j<INPUT;j++)
            Center[0][i][j] /= Pattern_Nos;
    }
    for(i=1;i<Pattern_Nos;i++)
        for(j=0;j<INPUT;j++){
            Weight[i][j] = Weight[0][j];
            for(k=0;k<INPUT;k++)
                Center[i][j][k] = Center[0][j][k];
        }
    /*
    for(i=0;i<Pattern_Nos;i++){
        for(j=0;j<INPUT;j++){
            //cout<<Weight[i][j]<<" ";
            for(k=0;k<INPUT;k++)
                cout<<Center[i][j][k]<<" ";
            cout<<"   ";
        }
        cout<<"\n";
    }
    */

}
```

```
void InitiateInput_Data(){
    float Ip;
    int i=0,j=0,set;
    //cout<<"\nInitiating Data . . .";
    //cout<<"\nPress any key to continue.";
    //_getch();
    std::ifstream MyCsvFile("Input.csv");
    if(!MyCsvFile)
        cout<<"Cant open file";
    std::string Line;
    while(getline(MyCsvFile,Line)){
        i=0;
        std::stringstream Line_stream(Line);
        std::string value;
        while(getline(Line_stream,value,',')){
            set=0;
            //cout<<"Record "<<record<<"\n";
            Ip=::atof(value.c_str());
            if(Ip){
                set=1;
                if(i==4){
                    Desired[j]=Ip;
                    //cout<<Desired[j];
                }
                else{
                    Input_X[j][i]=Ip;
                    //cout<<Neuron[j][i];
                }
            i++;
            }
            //cout<<"\n";
        }
        if(set)
            j++;
    }
}


void Print_Learn(){
    int i;
    std::ofstream Myfile;
    Myfile.open("Train_output.csv");
    Myfile<<",Inputs,,,,Weights\n";
    for(i=0;i<Pattern_Nos;i++)

Myfile<<Input_X[i][0]<<","<<Input_X[i][1]<<","<<Input_X[i][2]<<","<<Inpu
t_X[i][3]<<","<<Weight[i][0]<<","<<Weight[i][1]<<","<<Weight[i][2]<<","<
<Weight[i][3]<<"\n";
    Myfile.close();
}

void Print_Test(){
    int i;
    std::ofstream Myfile;
    Myfile.open("Test_output.csv");
    Myfile<<",Inputs,,,Outputs\n";
    for(i=0;i<Train_Nos;i++)

Myfile<<Input_X_Test[i][0]<<","<<Input_X_Test[i][1]<<","<<Input_X_Test[i
][2]<<","<<Input_X_Test[i][3]<<","<<Output_X[i]<<"\n";
    Myfile.close();
}

void Print_Output(){
    int i;
    std::ofstream Myfile;
    Myfile.open("Output.csv");
    Myfile<<",Inputs,,,Outputs\n";
    for(i=0;i<Pattern_Nos;i++)

Myfile<<Input_X[i][0]<<","<<Input_X[i][1]<<","<<Input_X[i][2]<<","<<Inpu
t_X[i][3]<<","<<Output_X[i]<<"\n";
    Myfile.close();
}


void Allocate_Memory(){
    int i,j;
    char ch;

    cout<<"\nNumber of Patterns = "<<Pattern_Nos<<"\nDo you want to
change (y/n)";
    cin>>ch;
    if(ch=='y'||ch=='Y'){
        cout<<"Enter Number of patterns : ";
        cin>>Pattern_Nos;
    }
    Train_Nos = Pattern_Nos/5;
    C_Nos = Pattern_Nos;
    cout<<"\nEpochs Limit = "<<Epochs_Max<<"\nDo you want to change
(y/n)";
    cin>>ch;
    if(ch=='y'||ch=='Y'){
        cout<<"Enter the Epochs Limit : ";
        cin>>Epochs_Max;
    }
    cout<<"\nFuzzy Tolerance Rate = "<<E_Toler<<"\nDo you want to change
(y/n)";
    cin>>ch;
    if(ch=='y'||ch=='Y'){
        cout<<"Enter the Fuzzy Tolerance Rate : ";
        cin>>E_Toler;
    }
    //cout<<"\nAllocating Memory\nPress any key to continue.";
    //_getch();
    V_Center = new float [C_Nos];
    U_Part = new float * [C_Nos];
    for(i=0;i<C_Nos;i++)
        U_Part[i] = new float [INPUT];
    U_Part1 = new float * [C_Nos];
    for(i=0;i<C_Nos;i++)
        U_Part1[i] = new float [INPUT];
    A_Mat = new float * [C_Nos];
    for(i=0;i<C_Nos;i++)
        A_Mat[i] = new float [INPUT];
    for(i=0;i<C_Nos;i++)
        for(j=0;j<INPUT;j++){
            U_Part[i][j]=1.0;
            if(i==j) A_Mat[i][j]=1.0;
            else A_Mat[i][j]=0.0;
        }
    Center = new float ** [Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++){
        Center [i] = new float * [INPUT];
        for(j=0;j<INPUT;j++)
            Center [i][j] = new float [INPUT];
    }
    Input_X = new float * [Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++)
        Input_X[i] = new float [INPUT];
    Weight = new float * [Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++)
        Weight[i] = new float [INPUT];
    Input_X_Test = new float * [Train_Nos];
    for(i=0;i<Train_Nos;i++)
        Input_X_Test[i] = new float [INPUT];
    PhiVal = new float * [Pattern_Nos];
    for(i=0;i<Pattern_Nos;i++)
        PhiVal [i] = new float [INPUT];
    Output_X = new float [Pattern_Nos];
    Desired = new float [Pattern_Nos];
    D_Max = new float [Pattern_Nos];
}

void Real_Phase(){
    //this->InitiateInput_Data();
    //this->Width_Calc();
    this->Activation_Calc();
    this->Output_Calc();
    this->Print_Output();
    cout<<"\nExecution Phase executed successfully\n";
}

void Testing_Phase(){
    //int i;
    this->Test_Initiate_Data();
    //this->Test_Width_Calc();
    this->Test_Activation_Calc();
    this->Test_Output_Calc();
    this->Print_Test();
    cout<<"\nTesting Phase executed successfully\n";
    /*
    cout<<"\n~~~~~~~~~~~~~~~~~~TEST OUTPUT~~~~~~~~~~~~~~~~~~~~~~~~\n";
    for(i=0;i<Train_Nos;i++)
        cout<<Input_X[i][0]<<"  "<<Input_X[i][1]<<"  "<<Input_X[i][2]<<"
"<<Input_X[i][3]<<"    "<<Output_X[i]<<"\n";
    */
}

void Initiate_phase(){
    this->Width_Calc();
    this->Activation_Calc();
    this->Output_Calc();
    this->ERate_Calc();
    this->ERMS_Calc();
    cout<<"\nEpochs = "<<Epochs<<"\tMean Square Error =
"<<E_Out<<"\tError RMS Value = "<<ERMS;
    Epochs++;
}

void Learning_Phase(){
    //int i;
    Epochs = 1;
    while(Epochs<=Epochs_Max){
        this->Initiate_phase();
        this->Update_Center();
        this->Update_Weight();
        //cout<<"\nEpochs = "<<Epochs<<"\tMean Square Error =
"<<E_Out<<"\tError RMS Value = "<<ERMS;
    }
    /*
    cout<<"\n\n~~~~~~~~~~~~~~~~~~~~learning Weight~~~~~~~~~~~~~~~~~\n";
    for(i=0;i<Pattern_Nos;i++)
        cout<<Weight[i][0]<<"  "<<Weight[i][1]<<"  "<<Weight[i][2]<<"
"<<Weight[i][3]<<"\n";
    */
    cout<<"\n\nLearning Phase executed successfully\n";
}

void Configuring_Phase(){
    this->Allocate_Memory();
    this->Initiate_Data();
    this->Center_Selection();
    this->Initiate_Weight();
}
};

int main(){
    int i;
    RBF_FCM obj;
    cout<<"\n\n~~~~~~~~~~~~~~~~WELCOME TO RADIAL BASIS NETWORK - FUZZY C
MEANS~~~~~~~~~~~~~~~~~\n\n";
    while(1){
```

```
        cout<<"\n\nKindly choose the option below :\n\n
1.Configuration \n    2.Learning Phase \n    3.Testing Phase \n
4.Execution \n    5.Exit\n";
        cin>>i;
        switch(i){
            case 1: {
                    obj.Configuring_Phase();
                    break;
            }
            case 2: {
                    obj.Learning_Phase();
                    break;
            }
            case 3:{
                    obj.Testing_Phase();
                    break;
            }
            case 4:{
                    obj.Real_Phase();
                    break;
            }
            default:exit(0);
        }
    }
        /*
        cout<<"\n";
        for(i=0;i<Pattern_Nos;i++)
        cout<<Input_X[i][0]<<" "<<Input_X[i][1]<<" "<<Input_X[i][2]<<"
"<<Input_X[i][3]<<" "<<Desired[i]<<"\n";
        */

    return 0;
}
```

APPENDIX:III

JAVA program for Inventory Management Lot sizing optimisation using ACO

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package advanced_aco;

/**
 *
 * @author Parv
 */
public class Advanced_ACO {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        InputData.Get_InputData();

        SolutionConstruction.generate_initial_solution();

        PheromoneUpdate pu = new PheromoneUpdate();

        TerminationCriteria tc = new TerminationCriteria();

        while (tc.TerminationCriteriaCheck() == false) {

            pu.initialize_fields();

            pu.pheromone_update();

            pu.store_best_solution();

            pu.store_worst_solution();

            SolutionConstruction.generate_solution();

            tc.Increment_Cycle_Completed_Byone();
        }

        pu.print_final_solution();

    }

}
```

Input data

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package advanced_aco;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

/**
 *
 * @author Parv
 */
public class InputData {

    public static int c;
    public static int p;
    public static int d;

    public static int[] O_lower;
    public static int[] O_upper;

    public static int[][] R;
    public static int[] G;
    public static int[] H;
    public static double[][] C;
    public static int[][] I;
    public static double[] A;
    public static double[] m;
    public static double M;

    public static int[][] PriceBreak_upper;

    public static void Get_InputData() {

        BufferedReader br = null;
        String line = "";
        String cvsSplitBy = ",";

        try {

            String csvFile = System.getProperty("user.dir") +
"/InputFiles/cpd.csv";

            br = new BufferedReader(new FileReader(csvFile));
            br.readLine();
            line = br.readLine();
            String[] cpd = line.split(cvsSplitBy);

            c = Integer.parseInt(cpd[0]);
            p = Integer.parseInt(cpd[1]);
            d = Integer.parseInt(cpd[2]);

            csvFile = System.getProperty("user.dir") +
"/InputFiles/VariableLimit.csv";

            br = new BufferedReader(new FileReader(csvFile));
            br.readLine();
            line = br.readLine();

            int[] O_lower_limit = new int[c];
            int[] O_upper_limit = new int[c];

            String[] variable_limit = line.split(cvsSplitBy);

            int l = 0, u = 0;

            for (int i = 0; i < variable_limit.length; i++) {
                if (i % 2 == 0) {
                    O_lower_limit[l++] =
Integer.parseInt(variable_limit[i]);
                } else {
                    O_upper_limit[u++] =
Integer.parseInt(variable_limit[i]);
                }
            }

            O_lower = O_lower_limit.clone();
            O_upper = O_upper_limit.clone();

            csvFile = System.getProperty("user.dir") +
"/InputFiles/InputData.csv";

            br = new BufferedReader(new FileReader(csvFile));
            br.readLine();
            line = br.readLine();
            String[] InputData = line.split(cvsSplitBy);
            int data = 0;

            int[][] RR = new int[c][p];

            for (int i = 0; i < c; i++) {
                for (int j = 0; j < p; j++) {
                    RR[i][j] = Integer.parseInt(InputData[data++]);
                }
            }

            R = RR.clone();

            int[] GG = new int[c];
            for (int i = 0; i < c; i++) {
                GG[i] = Integer.parseInt(InputData[data++]);
            }

            G = GG.clone();

            int[] HH = new int[c];

            for (int i = 0; i < c; i++) {
                HH[i] = Integer.parseInt(InputData[data++]);
            }

            H = HH.clone();

            double[][] CC = new double[c][d];

            for (int i = 0; i < c; i++) {
                for (int j = 0; j < d; j++) {
                    CC[i][j] = Double.parseDouble(InputData[data++]);
                }
            }

            C = CC.clone();

            int[][] II = new int[c][p];

            for (int i = 0; i < c; i++) {
                II[i][0] = Integer.parseInt(InputData[data++]);
            }

            I = II.clone();

            double[] AA = new double[p];

            for (int i = 0; i < p; i++) {
                AA[i] = Double.parseDouble(InputData[data++]);
            }

            A = AA.clone();

            double[] mm = new double[c];

            for (int i = 0; i < c; i++) {
                mm[i] = Double.parseDouble(InputData[data++]);
            }

            m = mm.clone();

            M = Double.parseDouble(InputData[data]);


            csvFile = System.getProperty("user.dir") +
"/InputFiles/PriceBreak.csv";
```

```
            br = new BufferedReader(new FileReader(csvFile));
            br.readLine();
            line = br.readLine();

            String[] PriceBreak = line.split(cvsSplitBy);

            data=0;

            int[][] PriceBreak_upper_limit  = new int[c][d-1];

            for (int i = 0; i < c; i++) {
                for (int j = 0; j < d-1; j++) {
                    PriceBreak_upper_limit[i][j] = 
Integer.parseInt(PriceBreak[data++]);
                }
            }

            PriceBreak_upper = PriceBreak_upper_limit.clone();

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {

            if (br != null) {
                try {
                    br.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

        }

        System.out.println("Input data files have been successfully 
read.");
        System.out.println();

    }

}
```

PHERMONE UPDATE

```
/*
 * To change this license header, choose License Headers in Project 
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package advanced_aco;
```

```
/**
 *
 * @author Parv
 */
public class PheromoneUpdate {

    double[] objective = new
double[SolutionConstruction.solution.length];

    public static double[] pheromone = new
double[SolutionConstruction.solution.length];

    int[][][] store_best_solution = new
int[TerminationCriteria.MaxCycle][InputData.c][InputData.p];

    double[] store_best_objective = new
double[TerminationCriteria.MaxCycle];

    double[] store_best_pheromone = new
double[TerminationCriteria.MaxCycle];

    int[][][] store_worst_solution = new
int[TerminationCriteria.MaxCycle][InputData.c][InputData.p];

    double[] store_worst_objective = new
double[TerminationCriteria.MaxCycle];

    double[] store_worst_pheromone = new
double[TerminationCriteria.MaxCycle];

    public static double sum_pheromone;

    public void initialize_fields() //initialize instance variables in
each cycle
    {
        int i;

        sum_pheromone = 0;

        for (i = 0; i < SolutionConstruction.solution.length; i++) {

            objective[i] = 0;

            pheromone[i] = 0;

        }

    }

    public void pheromone_update() // Update the pheromone in each
iteration (cycle)
    {
        int i, j, k;
        double[] total_ordering_cost = new
double[SolutionConstruction.solution.length];

        double[] total_holding_cost = new
double[SolutionConstruction.solution.length];

        double[] total_purchasing_cost = new
double[SolutionConstruction.solution.length];

        for (i = 0; i < SolutionConstruction.solution.length; i++) {
            for (j = 0; j < InputData.c; j++) {
                for (k = 0; k < InputData.p; k++) {
                    if (SolutionConstruction.solution[i][j][k] > 0) {
                        total_ordering_cost[i] = total_ordering_cost[i]
+ InputData.G[j];
                    }

                    if (k == 0) {
                        total_holding_cost[i] = total_holding_cost[i] +
InputData.H[j] * (InputData.I[j][k] +
SolutionConstruction.solution[i][j][k] - InputData.R[j][k] / 2);
                    } else {
                        if (k > 1) {
                            InputData.I[j][k - 1] = InputData.I[j][k -
2] + SolutionConstruction.solution[i][j][k - 2] - InputData.R[j][k - 2];
                        }
                        total_holding_cost[i] = total_holding_cost[i] +
InputData.H[j] * (InputData.I[j][k - 1] +
SolutionConstruction.solution[i][j][k - 1] - InputData.R[j][k - 1] +
SolutionConstruction.solution[i][j][k] - InputData.R[j][k] / 2);
                    }

                    int l;
                    for (l = 0; l < InputData.d - 1; l++) {
                        if (SolutionConstruction.solution[i][j][k] <=
InputData.PriceBreak_upper[j][l]) {
                            break;
                        }
                    }
                    total_purchasing_cost[i] = total_purchasing_cost[i]
+ (SolutionConstruction.solution[i][j][k] * InputData.C[j][l]);
                }
            }

            objective[i] = total_ordering_cost[i] +
total_holding_cost[i] + total_purchasing_cost[i];

            pheromone[i] = 1 / objective[i];

            sum_pheromone = sum_pheromone + pheromone[i];

        }
```

```
    }


public void store_best_solution() // Print the best solution in each
cycle
    {

        int i, j, k, maxIndex = 0;

        for (i = 0; i < pheromone.length; i++) {
            double newfitness = pheromone[i];
            if (newfitness > pheromone[maxIndex]) {
                maxIndex = i;
            }
        }

        System.out.println("Cycle {" +
TerminationCriteria.CycleCompleted + "}: Order qty:");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                System.out.format("%6d",
SolutionConstruction.solution[maxIndex][i][j]);

store_best_solution[TerminationCriteria.CycleCompleted][i][j] =
SolutionConstruction.solution[maxIndex][i][j];
            }
            System.out.println();
        }

        store_best_pheromone[TerminationCriteria.CycleCompleted] =
pheromone[maxIndex];
        store_best_objective[TerminationCriteria.CycleCompleted] =
objective[maxIndex];

        System.out.println("Cycle {" +
TerminationCriteria.CycleCompleted + "}: Decision variable Q: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                if
(store_best_solution[TerminationCriteria.CycleCompleted][i][j] == 0) {
                    System.out.format("%6d", 0);
                } else {
                    System.out.format("%6d", 1);
                }
            }
            System.out.println();
        }

        System.out.println("Cycle {" +
TerminationCriteria.CycleCompleted + "}: Decision variable B: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                boolean OnePrinted = false;
                for (k = 0; k < InputData.d - 1; k++) {
                    if
(store_best_solution[TerminationCriteria.CycleCompleted][i][j] <=
InputData.PriceBreak_upper[i][k]) {
                        if (OnePrinted == false) {
                            System.out.format("%2d", 1);
                            OnePrinted = true;
                        } else {
                            System.out.format("%2d", 0);
                        }
                    } else {
                        System.out.format("%2d", 0);
                    }

                }
                if (OnePrinted == false) {
                    System.out.format("%2d", 1);
                    System.out.print(" ");
                } else {
                    System.out.format("%2d", 0);
                    System.out.print(" ");
                }

            }
            System.out.println();
        }

        System.out.print("Cycle {" + TerminationCriteria.CycleCompleted
+ "}: Total cost: {" + objective[maxIndex] + "}");
        System.out.println();
        System.out.println();
    }


    public void store_worst_solution() // Print the worst solution in
each cycle
    {

        int i, j, minIndex = 0;

        for (i = 0; i < pheromone.length; i++) {
            double newfitness = pheromone[i];
            if (newfitness < pheromone[minIndex]) {
                minIndex = i;
            }
        }

        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {

store_worst_solution[TerminationCriteria.CycleCompleted][i][j] =
SolutionConstruction.solution[minIndex][i][j];
            }
        }
        store_worst_pheromone[TerminationCriteria.CycleCompleted] =
pheromone[minIndex];
        store_worst_objective[TerminationCriteria.CycleCompleted] =
objective[minIndex];

    }


    public void print_final_solution() // Print the final best and worst
solution
    {

        int i, j, k, maxIndex = 0;

        for (i = 0; i < store_best_pheromone.length; i++) {
            double newfitness = store_best_pheromone[i];
            if (newfitness > store_best_pheromone[maxIndex]) {
                maxIndex = i;
            }
        }

        System.out.println("Best final order qty: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                System.out.format("%6d",
store_best_solution[maxIndex][i][j]);
            }
            System.out.println();
        }

        System.out.println("Best final decision variable Q: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                if (store_best_solution[maxIndex][i][j] == 0) {
                    System.out.format("%6d", 0);
                } else {
                    System.out.format("%6d", 1);
                }
            }
            System.out.println();
        }

        System.out.println("Best final decision variable B: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                boolean OnePrinted = false;
                for (k = 0; k < InputData.d - 1; k++) {

                    if (store_best_solution[maxIndex][i][j] <=
InputData.PriceBreak_upper[i][k]) {
                        if (OnePrinted == false) {
                            System.out.format("%2d", 1);
                            OnePrinted = true;
                        } else {
                            System.out.format("%2d", 0);
                        }
                    } else {
                        System.out.format("%2d", 0);
                    }

                }
                if (OnePrinted == false) {
                    System.out.format("%2d", 1);
                    System.out.print(" ");
                } else {
                    System.out.format("%2d", 0);
                    System.out.print(" ");
                }

            }
            System.out.println();
        }
        System.out.println("Best final total cost: {" +
store_best_objective[maxIndex] + "}");
        System.out.println();

        int minIndex = 0;

        for (i = 0; i < store_worst_pheromone.length; i++) {
            double newfitness = store_worst_pheromone[i];
            if (newfitness < store_worst_pheromone[minIndex]) {
                minIndex = i;
            }
        }

        System.out.println("Worst final order qty: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                System.out.format("%6d",
store_worst_solution[minIndex][i][j]);
            }
            System.out.println();
        }

        System.out.println("Worst final decision variable Q: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                if (store_worst_solution[minIndex][i][j] == 0) {
                    System.out.format("%6d", 0);
                } else {
                    System.out.format("%6d", 1);
                }
            }
            System.out.println();
        }

        System.out.println("Worst final decision variable B: ");
```

```
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                boolean OnePrinted = false;
                for (k = 0; k < InputData.d - 1; k++) {

                    if (store_worst_solution[minIndex][i][j] <=
InputData.PriceBreak_upper[i][k]) {
                        if (OnePrinted == false) {
                            System.out.format("%2d", 1);
                            OnePrinted = true;
                        } else {
                            System.out.format("%2d", 0);
                        }
                    } else {
                        System.out.format("%2d", 0);
                    }

                }
                if (OnePrinted == false) {
                    System.out.format("%2d", 1);
                    System.out.print(" ");
                } else {
                    System.out.format("%2d", 0);
                    System.out.print(" ");
                }

            }

            System.out.println();
        }

        System.out.println("Worst final total cost: {" +
store_worst_objective[minIndex] + "}");
        System.out.println();

        double diff = store_worst_objective[minIndex] -
store_best_objective[maxIndex];
        System.out.println("Total cost difference: {" + diff + "}");
        System.out.println();

    }

}
TERMINATION  CRITERIA

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package advanced_aco;

/**
```

```
 *
 * @author Parv
 */
public class TerminationCriteria {

    public static int CycleCompleted;

    public static final int MaxCycle = 500;

    public static boolean TerminationCriteriaCheck() //Check whether
termination criteria met
    {

        if (CycleCompleted == MaxCycle) {
            return true;
        } else {
            return false;
        }

    }

    public static void Increment_Cycle_Completed_Byone() {

        CycleCompleted = CycleCompleted + 1;

    }

}
```

JAVA program for Inventory Management Lot sizing optimisation using GA

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package advanced_ga;

/**
 *
 * @author Parv
 */
public class Advanced_GA {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        InputData.Get_InputData();

        Population.generate_initial_population();

        Chromosome c = new Chromosome();

        TerminationCriteria tc = new TerminationCriteria();

        while (tc.TerminationCriteriaCheck() == false) {
            c.initialize_fields();
            c.calculate_fitness();
            c.store_best_chromosome();
            c.store_worst_chromosome();
            c.selection();
            c.crossover();
            c.mutation();
            c.replace();
            tc.Increment_Generation_Completed_Byone();
        }

        c.print_final_chromosome();

    }

}
```

Chromosome definition:

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package advanced_ga;

import java.util.Random;
```

```
/**
 *
 * @author Parv
 */
public class Chromosome {

    double[] objective = new double[Population.population.length];
    double[] fitness = new double[Population.population.length];

    int[][][] store_best_chromosome = new
int[TerminationCriteria.MaxGeneration][InputData.c][InputData.p];
    double[] store_best_objective = new
double[TerminationCriteria.MaxGeneration];
    double[] store_best_fitness = new
double[TerminationCriteria.MaxGeneration];

    int[][][] store_worst_chromosome = new
int[TerminationCriteria.MaxGeneration][InputData.c][InputData.p];
    double[] store_worst_objective = new
double[TerminationCriteria.MaxGeneration];
    double[] store_worst_fitness = new
double[TerminationCriteria.MaxGeneration];

    double sum_fitness;

    Random random = new Random();
    double random_number;

    int[] parent_selection = new int[Population.population.length];
    int[][][] offspring1 = new int[Population.population.length /
2][InputData.c][InputData.p];
    int[][][] offspring2 = new int[Population.population.length /
2][InputData.c][InputData.p];

    public void initialize_fields() //initialize instance variables in
each iteration
    {
        int i, j, k;
        sum_fitness = 0;
        random_number = 0;

        for (i = 0; i < Population.population.length; i++) {
            objective[i] = 0;
            fitness[i] = 0;
            parent_selection[i] = 0;
        }

        for (i = 0; i < Population.population.length / 2; i++) {
            for (j = 0; j < InputData.c; j++) {
                for (k = 0; k < InputData.p; k++) {
                    offspring1[i][j][k] = 0;
                    offspring2[i][j][k] = 0;
                }
            }
        }
    }

    public void calculate_fitness() // Evaluate the chromosomes in each
iteration (generation)
    {
        int i, j, k;
        double[] total_ordering_cost = new
double[Population.population.length];
        double[] total_holding_cost = new
double[Population.population.length];
        double[] total_purchasing_cost = new
double[Population.population.length];

        for (i = 0; i < Population.population.length; i++) {
            for (j = 0; j < InputData.c; j++) {
                for (k = 0; k < InputData.p; k++) {
                    if (Population.population[i][j][k] > 0) {
                        total_ordering_cost[i] = total_ordering_cost[i]
+ InputData.G[j];
                    }

                    if (k == 0) {
                        total_holding_cost[i] = total_holding_cost[i] +
InputData.H[j] * (InputData.I[j][k] + Population.population[i][j][k] -
InputData.R[j][k] / 2);
                    } else {
                        if (k > 1) {
                            InputData.I[j][k - 1] = InputData.I[j][k -
2] + Population.population[i][j][k - 2] - InputData.R[j][k - 2];
                        }
                        total_holding_cost[i] = total_holding_cost[i] +
InputData.H[j] * (InputData.I[j][k - 1] + Population.population[i][j][k
- 1] - InputData.R[j][k - 1] + Population.population[i][j][k] -
InputData.R[j][k] / 2);
                    }

                    int l;
                    for (l = 0; l < InputData.d - 1; l++) {
                        if (Population.population[i][j][k] <=
InputData.PriceBreak_upper[j][l]) {
                            break;
                        }
                    }
                    total_purchasing_cost[i] = total_purchasing_cost[i]
+ (Population.population[i][j][k] * InputData.C[j][l]);
                }
            }

            objective[i] = total_ordering_cost[i] +
total_holding_cost[i] + total_purchasing_cost[i];
            fitness[i] = 1 / objective[i];
            sum_fitness = sum_fitness + fitness[i];
        }

    }

    public void store_best_chromosome() // Print the best chromosome in
each iteration
    {

        int i, j, k, maxIndex = 0;

        for (i = 0; i < fitness.length; i++) {
            double newfitness = fitness[i];
            if (newfitness > fitness[maxIndex]) {
                maxIndex = i;
            }
        }

        System.out.println("Generation {" +
TerminationCriteria.GenerationCompleted + "}: Order qty:");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                System.out.format("%6d",
Population.population[maxIndex][i][j]);

store_best_chromosome[TerminationCriteria.GenerationCompleted][i][j] =
Population.population[maxIndex][i][j];
            }
            System.out.println();
        }

        store_best_fitness[TerminationCriteria.GenerationCompleted] =
fitness[maxIndex];
        store_best_objective[TerminationCriteria.GenerationCompleted] =
objective[maxIndex];

        System.out.println("Generation {" +
TerminationCriteria.GenerationCompleted + "}: Decision variable Q: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                if
(store_best_chromosome[TerminationCriteria.GenerationCompleted][i][j] ==
0) {
                    System.out.format("%6d", 0);
                } else {
                    System.out.format("%6d", 1);
                }
            }
            System.out.println();
        }

        System.out.println("Generation {" +
TerminationCriteria.GenerationCompleted + "}: Decision variable B: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                boolean OnePrinted = false;
                for (k = 0; k < InputData.d - 1; k++) {

                    if
(store_best_chromosome[TerminationCriteria.GenerationCompleted][i][j] <=
InputData.PriceBreak_upper[i][k]) {
                        if (OnePrinted == false) {
                            System.out.format("%2d", 1);
                            OnePrinted = true;
                        } else {
                            System.out.format("%2d", 0);
                        }
                    } else {
                        System.out.format("%2d", 0);
                    }

                }
                if (OnePrinted == false) {
                    System.out.format("%2d", 1);
                    System.out.print(" ");
                } else {
                    System.out.format("%2d", 0);
                    System.out.print(" ");
                }

            }
            System.out.println();
        }

        System.out.print("Generation {" +
TerminationCriteria.GenerationCompleted + "}: Total cost: {" +
objective[maxIndex] + "}");
        System.out.println();
        System.out.println();
    }

    public void store_worst_chromosome() // Print the worst chromosome
in each iteration
    {

        int i, j, minIndex = 0;

        for (i = 0; i < fitness.length; i++) {
            double newfitness = fitness[i];
            if (newfitness < fitness[minIndex]) {
                minIndex = i;
            }
        }

        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
```

```
store_worst_chromosome[TerminationCriteria.GenerationCompleted][i][j] =
Population.population[minIndex][i][j];
            }
        }

        store_worst_fitness[TerminationCriteria.GenerationCompleted] =
fitness[minIndex];
        store_worst_objective[TerminationCriteria.GenerationCompleted] =
objective[minIndex];

    }

    public void selection() // Apply roulette wheel selection operation
in each iteration
    {
        double sum_of_fitness;
        int i, j;

        for (i = 0; i < Population.population.length; i++) {
            sum_of_fitness = 0;
            random_number = random.nextDouble();
            random_number = random_number * sum_fitness;

            for (j = 0; j < Population.population.length; j++) {
                sum_of_fitness = sum_of_fitness + fitness[j];

                if (sum_of_fitness > random_number) {
                    parent_selection[i] = j;
                    break;
                }
            }
        }
    }

    public void crossover() // Apply single point crossover operation in
each iteration
    {
        int i, j, k = 0, l;

        int[][][] parent1
               = new int[Population.population.length /
2][InputData.c][InputData.p];

        int[][][] parent2
               = new int[Population.population.length /
2][InputData.c][InputData.p];

        double crossoverProbability = 0.9;

        for (i = 0; i < Population.population.length / 2; i++) {
            for (j = 0; j < InputData.c; j++) {
                for (l = 0; l < InputData.p; l++) {
                    parent1[i][j][l] =
Population.population[parent_selection[i]][j][l];
                }
            }
        }

        for (i = Population.population.length / 2; i <
Population.population.length; i++) {
            for (j = 0; j < InputData.c; j++) {
                for (l = 0; l < InputData.p; l++) {

                    parent2[k][j][l] =
Population.population[parent_selection[i]][j][l];
                }
            }

            k = k + 1;
        }

        int sum_offspring1[] = new int[InputData.c];
        int sum_offspring2[] = new int[InputData.c];

        for (i = 0; i < Population.population.length / 2; i++) {
            random_number = random.nextDouble();

            if (random_number <= crossoverProbability) {
                int crossoverPoint = InputData.p - 1;

                for (j = 0; j < InputData.c; j++) {
                    for (l = 0; l < crossoverPoint; l++) {
                        offspring1[i][j][l] = parent1[i][j][l];
                        offspring2[i][j][l] = parent2[i][j][l];
                        sum_offspring1[j] = sum_offspring1[j] +
offspring1[i][j][l];
                        sum_offspring2[j] = sum_offspring2[j] +
offspring2[i][j][l];
                    }
                }

                for (j = 0; j < InputData.c; j++) {
                    for (l = crossoverPoint; l < InputData.p; l++) {
                        offspring1[i][j][l] = parent2[i][j][l];
                        offspring2[i][j][l] = parent1[i][j][l];
                        sum_offspring1[j] = sum_offspring1[j] +
offspring1[i][j][l];
                        sum_offspring2[j] = sum_offspring2[j] +
offspring2[i][j][l];
                    }
                    int diff1 = sum_offspring1[j] -
Population.TotalRCompWise[j];
                    int diff2 = sum_offspring2[j] -
Population.TotalRCompWise[j];
                    if (diff1 < 0) {
                        offspring1[i][j][InputData.p - 1] =
offspring1[i][j][InputData.p - 1] - diff1;
                    } else {
                        offspring1[i][j][InputData.p - 1] =
offspring1[i][j][InputData.p - 1] - diff1;
                    }

                    if (diff2 < 0) {
                        offspring2[i][j][InputData.p - 1] =
offspring2[i][j][InputData.p - 1] - diff2;
                    } else {
                        offspring2[i][j][InputData.p - 1] =
offspring2[i][j][InputData.p - 1] - diff2;
                    }

                }

            } else {

                for (j = 0; j < InputData.c; j++) {
                    for (l = 0; l < InputData.p; l++) {
                        offspring1[i][j][l] = parent1[i][j][l];
                        offspring2[i][j][l] = parent2[i][j][l];
                    }
                }

            }
        }
    }

    public void mutation() // Apply bit-wise mutation operation in each
iteration
    {
        int mutation_value;
        double mutation_probability = 0.4;
        int i, j, k = 0, l;
        for (i = 0; i < Population.population.length / 2; i++) {
            random_number = random.nextDouble();
            if (random_number <= mutation_probability) {

                for (j = 0; j < InputData.c; j++) {

                    int selected_period = random.nextInt(InputData.p);
                    if (selected_period == 0) {
                        if ((offspring1[i][j][selected_period] -
InputData.R[j][selected_period] + 1) > 0) {
                            mutation_value =
random.nextInt(offspring1[i][j][selected_period] -
InputData.R[j][selected_period] + 1);
                            offspring1[i][j][selected_period] =
offspring1[i][j][selected_period] - mutation_value;
                            offspring1[i][j][selected_period + 1] =
offspring1[i][j][selected_period + 1] + mutation_value;
                        }
                    } else if (selected_period == InputData.p - 1) {
                        if ((offspring1[i][j][selected_period] + 1) > 0)
{
                            mutation_value =
random.nextInt(offspring1[i][j][selected_period] + 1);
                            offspring1[i][j][selected_period] =
offspring1[i][j][selected_period] - mutation_value;
                            offspring1[i][j][0] = offspring1[i][j][0] +
mutation_value;
                        }
                    } else {
                        int max = 0;
                        for (int n = 0; n <= selected_period; n++) {
                            max = max + offspring1[i][j][n] -
InputData.R[j][n];
                        }
                        if (max <= offspring1[i][j][selected_period]) {
                            if ((max + 1) > 0) {
                                mutation_value = random.nextInt(max +
1);
                                offspring1[i][j][selected_period] =
offspring1[i][j][selected_period] - mutation_value;
                                offspring1[i][j][selected_period + 1] =
offspring1[i][j][selected_period + 1] + mutation_value;
                            }
                        } else {
                            if ((offspring1[i][j][selected_period] + 1)
> 0) {
                                mutation_value =
random.nextInt(offspring1[i][j][selected_period] + 1);
                                offspring1[i][j][selected_period] =
offspring1[i][j][selected_period] - mutation_value;
                                offspring1[i][j][selected_period + 1] =
offspring1[i][j][selected_period + 1] + mutation_value;

                            }
                        }
                    }

                    selected_period = random.nextInt(InputData.p);
                    if (selected_period == 0) {
                        if ((offspring2[i][j][selected_period] -
InputData.R[j][selected_period] + 1) > 0) {
                            mutation_value =
random.nextInt(offspring2[i][j][selected_period] -
InputData.R[j][selected_period] + 1);
                            offspring2[i][j][selected_period] =
offspring2[i][j][selected_period] - mutation_value;
                            offspring2[i][j][selected_period + 1] =
offspring2[i][j][selected_period + 1] + mutation_value;
                        }
                    } else if (selected_period == InputData.p - 1) {
                        if ((offspring2[i][j][selected_period] + 1) > 0)
{
```

```
                            mutation_value =
random.nextInt(offspring2[i][j][selected_period] + 1);
                            offspring2[i][j][selected_period] =
offspring2[i][j][selected_period] - mutation_value;
                            offspring2[i][j][0] = offspring2[i][j][0] +
mutation_value;
                        }
                    } else {
                        int max = 0;
                        for (int n = 0; n <= selected_period; n++) {
                            max = max + offspring2[i][j][n] -
InputData.R[j][n];
                        }
                        if (max <= offspring2[i][j][selected_period]) {
                            if ((max + 1) > 0) {
                                mutation_value = random.nextInt(max +
1);

                                offspring2[i][j][selected_period] =
offspring2[i][j][selected_period] - mutation_value;
                                offspring2[i][j][selected_period + 1] =
offspring2[i][j][selected_period + 1] + mutation_value;
                            }
                        } else {
                            if ((offspring2[i][j][selected_period] + 1)
> 0) {
                                mutation_value =
random.nextInt(offspring2[i][j][selected_period] + 1);
                                offspring2[i][j][selected_period] =
offspring2[i][j][selected_period] - mutation_value;
                                offspring2[i][j][selected_period + 1] =
offspring2[i][j][selected_period + 1] + mutation_value;

                            }
                        }

                    }

                }

            }

            for (j = 0; j < InputData.c; j++) {
                for (l = 0; l < InputData.p; l++) {
                    Population.new_population[k][j][l] =
offspring1[i][j][l];
                }
            }
            k = k + 1;

            for (j = 0; j < InputData.c; j++) {
                for (l = 0; l < InputData.p; l++) {
                    Population.new_population[k][j][l] =
offspring2[i][j][l];
                }
            }
            k = k + 1;
        }
    }

    public void replace() { //Replace old population by new population
in each iteration

        int h = 0;

        while (h < Population.population_size) {
            double[] TotalBudgetPerWise = new double[InputData.p];
            boolean IsValidPopulation = true;
            for (int j = 0; j < InputData.p; j++) {
                for (int i = 0; i < InputData.c; i++) {
                    int k;
                    for (k = 0; k < InputData.d - 1; k++) {
                        if (Population.new_population[h][i][j] <=
InputData.PriceBreak_upper[i][k]);
                        {
                            break;
                        }
                    }
                    TotalBudgetPerWise[j] = TotalBudgetPerWise[j] +
(Population.new_population[h][i][j] * InputData.C[i][k]);
                }

                if (TotalBudgetPerWise[j] > InputData.A[j]) {
                    IsValidPopulation = false;
                    break;
                }
            }

            if (IsValidPopulation == true) {
                double[] TotalAreaPerWise = new double[InputData.p];
                for (int j = 0; j < InputData.p; j++) {
                    for (int i = 0; i < InputData.c; i++) {

                        if (j == 0) {
                            TotalAreaPerWise[j] = TotalAreaPerWise[j] +
InputData.m[i] * (InputData.I[i][j] +
Population.new_population[h][i][j]);
                        } else {
                            if (j > 1) {
                                InputData.I[i][j - 1] = InputData.I[i][j
- 2] + Population.new_population[h][i][j - 2] - InputData.R[i][j - 2];
                            }
                            TotalAreaPerWise[j] = TotalAreaPerWise[j] +
InputData.m[i] * (InputData.I[i][j - 1] +
Population.new_population[h][i][j - 1] - InputData.R[i][j - 1] +
Population.new_population[h][i][j]);
                        }

                    }
                    if (TotalAreaPerWise[j] > InputData.M) {
                        IsValidPopulation = false;
                        break;
                    }
                }
            }

            if (IsValidPopulation == true) {
                for (int i = 0; i < InputData.c; i++) {
                    if (Population.new_population[h][i][InputData.p - 1]
< 0) {
                        IsValidPopulation = false;
                        break;
                    }
                }
            }

            if (IsValidPopulation == true) {
                for (int i = 0; i < InputData.c; i++) {
                    for (int j = 0; j < InputData.p; j++) {
                        Population.population[h][i][j] =
Population.new_population[h][i][j];
                    }
                }
            }

            h++;
        }
    }

    public void print_final_chromosome() // Print the final best and
worst chromosome as a solution
    {

        int i, j, k, maxIndex = 0;

        for (i = 0; i < store_best_fitness.length; i++) {
            double newfitness = store_best_fitness[i];
            if (newfitness > store_best_fitness[maxIndex]) {
                maxIndex = i;
            }
        }

        System.out.println("Best final order qty: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                System.out.format("%6d",
store_best_chromosome[maxIndex][i][j]);
            }
            System.out.println();
        }

        System.out.println("Best final decision variable Q: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                if (store_best_chromosome[maxIndex][i][j] == 0) {
                    System.out.format("%6d", 0);
                } else {
                    System.out.format("%6d", 1);
                }
            }
            System.out.println();
        }

        System.out.println("Best final decision variable B: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                boolean OnePrinted = false;
                for (k = 0; k < InputData.d - 1; k++) {

                    if (store_best_chromosome[maxIndex][i][j] <=
InputData.PriceBreak_upper[i][k]) {
                        if (OnePrinted == false) {
                            System.out.format("%2d", 1);
                            OnePrinted = true;
                        } else {
                            System.out.format("%2d", 0);
                        }
                    } else {
                        System.out.format("%2d", 0);
                    }

                }
                if (OnePrinted == false) {
                    System.out.format("%2d", 1);
                    System.out.print(" ");
                } else {
                    System.out.format("%2d", 0);
                    System.out.print(" ");
                }

            }
            System.out.println();
        }
        System.out.println("Best final total cost: {" +
store_best_objective[maxIndex] + "}");
        System.out.println();

        int minIndex = 0;

        for (i = 0; i < store_worst_fitness.length; i++) {
            double newfitness = store_worst_fitness[i];
            if (newfitness < store_worst_fitness[minIndex]) {
                minIndex = i;
            }
        }
        System.out.println("Worst final order qty: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                System.out.format("%6d",
store_worst_chromosome[minIndex][i][j]);
            }
            System.out.println();
        }

        System.out.println("Worst final decision variable Q: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                if (store_worst_chromosome[minIndex][i][j] == 0) {
                    System.out.format("%6d", 0);
                } else {
                    System.out.format("%6d", 1);
                }
            }
            System.out.println();
        }

        System.out.println("Worst final decision variable B: ");
        for (i = 0; i < InputData.c; i++) {
            for (j = 0; j < InputData.p; j++) {
                boolean OnePrinted = false;
                for (k = 0; k < InputData.d - 1; k++) {

                    if (store_worst_chromosome[minIndex][i][j] <=
InputData.PriceBreak_upper[i][k]) {
                        if (OnePrinted == false) {
                            System.out.format("%2d", 1);
                            OnePrinted = true;
                        } else {
                            System.out.format("%2d", 0);
                        }
                    } else {
                        System.out.format("%2d", 0);
                    }

                }
                if (OnePrinted == false) {
                    System.out.format("%2d", 1);
                    System.out.print(" ");
                } else {
                    System.out.format("%2d", 0);
                    System.out.print(" ");
                }

            }
            System.out.println();
        }
        System.out.println("Worst final total cost: {" +
store_worst_objective[minIndex] + "}");
        System.out.println();
        double diff = store_worst_objective[minIndex] - 
store_best_objective[maxIndex];
        System.out.println("Total cost difference: {" + diff + "}");
        System.out.println();
    }

}
```

INPUT DATA

```
/*
 * To change this license header, choose License Headers in Project 
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package advanced_ga;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

/**
 *
 * @author Parv
 */
public class InputData {

    public static int c;
    public static int p;
    public static int d;

    public static int[] O_lower;
    public static int[] O_upper;

    public static int[][] R;
    public static int[] G;
    public static int[] H;
    public static double[][] C;
    public static int[][] I;
    public static double[] A;
    public static double[] m;
    public static double M;

    public static int[][] PriceBreak_upper;

    public static void Get_InputData() {

        BufferedReader br = null;
        String line = "";
        String cvsSplitBy = ",";

        try {

            String csvFile = System.getProperty("user.dir") +
"/InputFiles/cpd.csv";

            br = new BufferedReader(new FileReader(csvFile));
            br.readLine();
            line = br.readLine();

            String[] cpd = line.split(cvsSplitBy);

            c = Integer.parseInt(cpd[0]);
            p = Integer.parseInt(cpd[1]);
            d = Integer.parseInt(cpd[2]);

            csvFile = System.getProperty("user.dir") +
"/InputFiles/VariableLimit.csv";

            br = new BufferedReader(new FileReader(csvFile));
            br.readLine();
            line = br.readLine();

            int[] O_lower_limit = new int[c];
            int[] O_upper_limit = new int[c];

            String[] variable_limit = line.split(cvsSplitBy);

            int l = 0, u = 0;

            for (int i = 0; i < variable_limit.length; i++) {
                if (i % 2 == 0) {
                    O_lower_limit[l++] =
Integer.parseInt(variable_limit[i]);
                } else {
                    O_upper_limit[u++] =
Integer.parseInt(variable_limit[i]);
                }
            }

            O_lower = O_lower_limit.clone();
            O_upper = O_upper_limit.clone();

            csvFile = System.getProperty("user.dir") +
"/InputFiles/InputData.csv";

            br = new BufferedReader(new FileReader(csvFile));
            br.readLine();
            line = br.readLine();
            String[] InputData = line.split(cvsSplitBy);
            int data = 0;

            int[][] RR = new int[c][p];
```

```
for (int i = 0; i < c; i++) {
    for (int j = 0; j < p; j++) {
        RR[i][j] = Integer.parseInt(InputData[data++]);
    }
}

R = RR.clone();

int[] GG = new int[c];

for (int i = 0; i < c; i++) {
    GG[i] = Integer.parseInt(InputData[data++]);
}

G = GG.clone();

int[] HH = new int[c];

for (int i = 0; i < c; i++) {
    HH[i] = Integer.parseInt(InputData[data++]);
}

H = HH.clone();

double[][] CC = new double[c][d];

for (int i = 0; i < c; i++) {
    for (int j = 0; j < d; j++) {
        CC[i][j] = Double.parseDouble(InputData[data++]);
    }
}

C = CC.clone();

int[][] II = new int[c][p];

for (int i = 0; i < c; i++) {
    II[i][0] = Integer.parseInt(InputData[data++]);
}

I = II.clone();

double[] AA = new double[p];

for (int i = 0; i < p; i++) {
    AA[i] = Double.parseDouble(InputData[data++]);
}

A = AA.clone();

double[] mm = new double[c];

for (int i = 0; i < c; i++) {
```

```
                mm[i] = Double.parseDouble(InputData[data++]);
            }

            m = mm.clone();

            M = Double.parseDouble(InputData[data]);


            csvFile = System.getProperty("user.dir") +
"/InputFiles/PriceBreak.csv";

            br = new BufferedReader(new FileReader(csvFile));
            br.readLine();
            line = br.readLine();

            String[] PriceBreak = line.split(cvsSplitBy);

            data=0;

            int[][] PriceBreak_upper_limit  = new int[c][d-1];

            for (int i = 0; i < c; i++) {
                for (int j = 0; j < d-1; j++) {
                    PriceBreak_upper_limit[i][j] =
Integer.parseInt(PriceBreak[data++]);
                }
            }

            PriceBreak_upper = PriceBreak_upper_limit.clone();

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {

            if (br != null) {
                try {
                    br.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

        }

        System.out.println("Input data files have been successfully
read.");
        System.out.println();

    }

}
```

POPULATION

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package advanced_ga;

import java.util.Random;

/**
 *
 * @author Parv
 */
public class Population {

    public static final int population_size = 5000;

    public static int[][][] population = new
int[population_size][InputData.c][InputData.p];
    public static int[][][] new_population = new
int[population_size][InputData.c][InputData.p];
    public static int[] TotalRCompWise = new int[InputData.c];

    public static void generate_initial_population() {

        int h = 0;

        int[] TotalOrderQty = new int[InputData.c];

        for (int i = 0; i < InputData.c; i++) {

            for (int j = 0; j < InputData.p; j++) {
                TotalRCompWise[i] = TotalRCompWise[i] +
InputData.R[i][j];
            }

        }

        Random random = new Random();

        while (h < population_size) {

            for (int i = 0; i < InputData.c; i++) {

                TotalOrderQty[i] = 0;

                for (int j = 0; j < InputData.p; j++) {
                    if (j == InputData.p - 1) {
                        population[h][i][j] = TotalRCompWise[i] -
TotalOrderQty[i];
                    } else {
                        if (j == 0) {
                            population[h][i][j] =
random.nextInt((TotalRCompWise[i] + 1) - InputData.R[i][j]) +
InputData.R[i][j];
                            TotalOrderQty[i] = TotalOrderQty[i] +
population[h][i][j];
                        } else {
                                if(TotalRCompWise[i] - TotalOrderQty[i]
== 0)
                                {
                                    population[h][i][j]=0;
                                }
                                else
                                {
                                    int min_second_term = 0;
                                    for(int p=0;p<j;p++)
                                    {
                                        min_second_term =
min_second_term + (population[h][i][p]-InputData.R[i][p]);
                                    }
                                    int min = InputData.R[i][j]-
min_second_term;
                                    if(min<0)
                                    {
                                        min=0;
                                    }
                                    int max = TotalRCompWise[i] -
TotalOrderQty[i];
                                    population[h][i][j] =
random.nextInt((max + 1) - min) + min;
                                    TotalOrderQty[i] = TotalOrderQty[i]
+ population[h][i][j];
                                }
                        }

                    }
                }
            }

            double[] TotalBudgetPerWise = new double[InputData.p];

            boolean IsValidPopulation = true;

            for (int j = 0; j < InputData.p; j++) {

                for (int i = 0; i < InputData.c; i++) {
                    int k;

                    for (k = 0; k < InputData.d - 1; k++) {
                        if (population[h][i][j] <=
InputData.PriceBreak_upper[i][k]) {
                            break;
                        }

                    }

                    TotalBudgetPerWise[j] = TotalBudgetPerWise[j] +
(population[h][i][j] * InputData.C[i][k]);
                }

                if (TotalBudgetPerWise[j] > InputData.A[j]) {
                    IsValidPopulation = false;
                    break;
                }

            }

            if (IsValidPopulation == true) {

                double[] TotalAreaPerWise = new double[InputData.p];

                for (int j = 0; j < InputData.p; j++) {

                    for (int i = 0; i < InputData.c; i++) {

                        if (j == 0) {
                            TotalAreaPerWise[j] = TotalAreaPerWise[j] +
InputData.m[i] * (InputData.I[i][j] + population[h][i][j]);
                        } else {

                            if (j > 1) {
                                InputData.I[i][j - 1] = InputData.I[i][j
- 2] + population[h][i][j - 2] - InputData.R[i][j - 2];
                            }

                            TotalAreaPerWise[j] = TotalAreaPerWise[j] +
InputData.m[i] * (InputData.I[i][j - 1] + population[h][i][j - 1] -
InputData.R[i][j - 1] + population[h][i][j]);
                        }

                    }

                    if (TotalAreaPerWise[j] > InputData.M) {
                        IsValidPopulation = false;
                        break;
                    }
                }
            }

            if (IsValidPopulation == true) {
                h++;
            }
```

```
        }

    }

}
```

TERMINATION CRITERIA

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package advanced_ga;

/**
 *
 * @author Parv
 */
public class TerminationCriteria {

    public static int GenerationCompleted;

    public static final int MaxGeneration = 500;

    public static boolean TerminationCriteriaCheck() //Check whether
termination criteria met
    {

        if (GenerationCompleted == MaxGeneration) {
            return true;
        } else {
            return false;
        }

    }

    public static void Increment_Generation_Completed_Byone() {

        GenerationCompleted = GenerationCompleted + 1;

    }

}
```

List of Publications based on PhD Research Work

Sl. No.	Title of the Paper	Authors	Name of the journal/ Conference/Symposium, Vol., No., Pages	Month & Year of publication	Category
1	Population based Meta-heuristic Algorithm approach for Analysis of Multi item Multi period Procurement Lot sizing Problem	Prasanna kumar, Dr.Mervin Herbert, Dr.SrikanthRao	Hindawi Advances in Operations Research Volume 2017, Article ID 3601217. https://doi.org/10.1155/2017/3601217	20-12-2017	Journal paper
2	Solution of multi-item multi-period inventory control problem using novel meta-heuristics and hybrid method	Prasanna kumar, Dr.Mervin Herbert, Dr.SrikanthRao	Journal of Adv Research in Dynamical & Control Systems,ISSN 1943-023X Vol. 9, No. 4, 2017Page no.150-160 (Scopus Indexed)	June 2017	Journal paper
3	Genetic algorithm approach for analysis of multi item multi period procurement lot sizing problem	Prasanna Kumar, Dr.Mervin Herbert, Dr.SrikanthRao	International Journal of Management (IJM) Volume 6, Issue 12, Dec 2015, pp. 50-58, Article ID:IJM_06_12_005 ISSN Print: 0976-6502 and ISSN Online: 0976-6510	Dec 2015	Journal paper
4	Demand forecasting using Artificial Neural Network based on different learning methods: Comparative Analysis	Prasanna kumar, Dr.Mervin Herbert, Dr.SrikanthRao	International Journal for Research in Applied Science and Engineering Technology Vol. 2 Issue IV, April 2014	April 2014	Journal paper
5	AI Technique Applications in Inventory Management A Review and Analysis of Literature”	Prasanna kumar, Dr.Mervin Herbert, Dr.SrikanthRao	International Journal of Business and Management Tomorrow,.vol 3, No. 5	May 2013	Journal paper

Sl. No.	Title of the Paper	Authors	Name of the journal/ Conference/Symposium, Vol., No., Pages	Month & Year of publication	Category
6	Solution for Supply Chain Management Challenges- An Approach of Integrated Application of AI techniques	Prasanna kumar, Dr.Mervin Herbert, Dr.SrikanthRao	submitted for SCI indexed journal Journal of Computer Information Systems (Under review)	Submitted on 15 Sept. 2017	Journal paper Under review
1c	Application of Artificial Intelligence technique for demand forecasting	Prasanna kumar, Dr.Mervin Herbert, Dr.SrikanthRao	International Conference on Mechanical And Production Engineering (ICMPE) on 06thJuly, 2014 at Bangalore Organized by Institute of Technology & Research	06thJuly, 2014	Conference
2c	Artificial Neural Network Approach for Industrial Demand Forecast	Prasanna kumar, Dr.Mervin Herbert, Dr.SrikanthRao	International Conference on Mechanical Aeronautical And Production Engineering on 25 th April 2015 at Singapore International Institute of Engineers and Researchers	25th April 2015	Conference

RESUME

The author Prasanna Kumar was born on 13th March 1966 at Surathkal, Mangalore. He obtained his primary education in Government Primary School, Katipalla, Krishnapura. He completed his secondary and higher secondary education in Vidyadayinee High school and Govindadas College , Surathkal, respectively. He secured his Bachelors degree in Mechanical Engineering from Bangalore University in the year 1988 with First Class (Distinction). He worked with M/s Harita Group of companies, sister concern of TVS group, Hosur, Bangalore, for a brief period of one year as Graduate Engineer Trainee and then worked as Assistant Executive Engineer with Oil and Natural Gas Corporation for 3 years.

He later joined Valve Chem Industires, subsidiary of Shalimar valves Ltd, Mumbai as Project Engineer. He obtained his Masters degree in Business Management from Mangalore University in the year 2003 securing First Class with Distinction.

The author has published a part of his research work in the form of papers in International Conferences and four papers in an International Journal. The papers in International Conferences have been presented at Bangalore, organized by Institute of Technology and Research and at Singapore, organized by International Institute of Engineers and Researchers. The author also has submitted two more papers to an International Journal which is presently under review.

The author has been associated with supply chain management, inventory management and control. He has expertise in the use ERP software like SAP for application in Material Management. His research interests include Neural Network modeling of demand forecast and application of AI technique for multi objective optimization in Inventory management.