Development of Neural Augmented Ant Colony Optimization (NaACO) Technique for Scheduling Problems

Author

MUHAMMAD UMER

2011-NUST-DirPhD-Mfr-E-12

Supervisor

DR RIAZ AHMAD

SCHOOL OF MECHANICAL AND MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

DECEMBER, 2014

Development of Neural Augmented Ant Colony Optimization (NaACO) Technique for Scheduling Problems

Author

MUHAMMAD UMER

2011-NUST-DirPhD-Mfr-E-12

A thesis submitted in partial fulfillment of the requirements for the degree of

PhD Industrial and Manufacturing Engineering

Thesis Supervisor:

DR RIAZ AHMAD

Thesis Supervisor's Signature: ______________________________________

SCHOOL OF MECHANICAL AND MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

DECEMBER, 2014

Declaration

I certify that this research work titled "*Development of Neural Augmented Ant Colony Optimization (NaACO) Technique for Scheduling Problems"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Muhammad Umer

2011-NUST-DirPhD-Mfr-E-12

Language Correctness Certificate

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Muhammad Umer

2011-NUST-DirPhD-Mfr-E-12

Dr. Riaz Ahmad

Copyright Statement

Acknowledgements

The greatest acknowledgement of my endeavors is due to the Taufeeque provided by ALLAH Subhana wa Ta'allah. My endeavors are imbued through the mercy of my Prophet (Sallah O Alaih'e Wassalam) as said in Holy Quran:

"And We have sent you O MUHAMMAD (Sallah O Alaih'e Wassalam) not but as a mercy for Alameen (all that exists)…. (Sur'ah AL-Ambiah verse 108)".

I hold in greatest esteem and acknowledge the guidance and supervision in life provided by my teacher His Respected Abdus Samad (Agha Jaan), my Mohsin from Lahore and my Mohsin's Mohsin from beyond Lahore.I dedicate this thesis to my Father Late Dr. Muhammad Shamim and my Mother Anjum Ara Alvi. I also dedicate this effort to my uncle Mr. Raffique Ahmad who always encouraged me to take risks. I greatly and firstly acknowledge my advisor Dr. Riaz Ahmad for his patience, perseverance and guidance, along with my research committee members Dr. Shahid Ikramullah, Dr. Imran Chaudhary and Dr. Amir Farhan. They have been my hinge pin and motivation to go through this endeavor. My admiration is for all my family members who were with me on this mission and who gave me time at their discretion to indulge in research and to publish.I couldn't have done it without the very able and timely guidance of Mr. Naveed Zafar. I salute my team (baber, hira, umaer) for giving me time, effort and hard work. My special thanks to Saad Siddique for being there for me every time. My gratitude to Fakhar for coordination.I am especially grateful to Mr. Mian Tahir Aftab for exposing me to Ant Colony Optimization technique, Mr. Waqas for giving me insights for my second case study, Mr. Nasir Raja for contributing in the application of the technique.In the end I am specially honored to be encouraged to pursue my work by the inventor of this technique **Dr. Marcus Dorigo**, and **Dr. Thomas Stutelzle** who commented in the following phrases in response to my endeavors (email as received):

Dear Muhammad,

Thanks for your kind email.When you will be ready with a paper you might want to consider submitting it to the ANTS conference:http://iridia.ulb.ac.be/ants2014/Regards,**Marco Dorigo.**

Dear Muhammad Umer,

Thanks for the information on the work in your group and I'm happy to hear that there are also people working on ACO in Pakistan. ***Thomas Stutelzle***

I dedicate my efforts of four years and this PhD thesis to the young martyrs of Peshawar (16/12/14). We are not the same without your smiles now. Pakistan will not be the same without you.

Abstract

This dissertation focuses on the development of an intelligent prone methodology for efficient and effective handling of scheduling problems. In industrial concerns the issues/problems related to resource scheduling arise from abrupt and sudden demand and want patterns due to clustered and unbalanced supply of resources and assets. A novel technique in this respect has been developed and implemented on cases from industry. This technique takes into stride the efficiency demonstrated by various PSO (Particle Swarm Optimization) inspired techniques, combined with the intelligent prone ANNs (Artificial Neural Networks). Novelty and uniqueness is demonstrated through amalgamation of these approaches to introduce a term: **NaACO (Neural Augmented ACO).** This formulation is done under the umbrella of introduction of yet another unique approach i.e **i-ACO** (Intelligent ACO) theme through **Neu(Tau) or Neuτ.**This thesis also focuses on how ACO takes into account and absorbs the neural aspect of supervised and unsupervised learning. The intention of this research is to come up with a unique, customized and yet efficient way to handle the problems of the industry under given limitations and constraints. A complete model for this approach is built and for the application of the model a high technology aviation maintenance industry (case study I) is selected along with a medium technology manufacturing setup (case study II). The usage of ACO meta-heuristic is taken as an ideal reference point with which every problem set can be converged towards a best fit solution. Subsequently ANN is used to come up with a combitorial dialog box to prompt for the inputs and evaluate the outputs of the given problem. The thesis contributes to current research by introducing **NaACO, i-ACO** through intelligent scheduling (hence introducing **neuτ**); which proposes many solutions of the existing problems. The discussion and conclusions part at the end summarizes the research and the future areas of research are also elaborated to assist and appreciate future researchers who are interested to endeavor in this field and related applications.

Key Words: *Ant Colony Optimization (ACO), Artificial Neural Networks (ANNs), Neural Augmented ACO (NaACO)*

Table of Contents

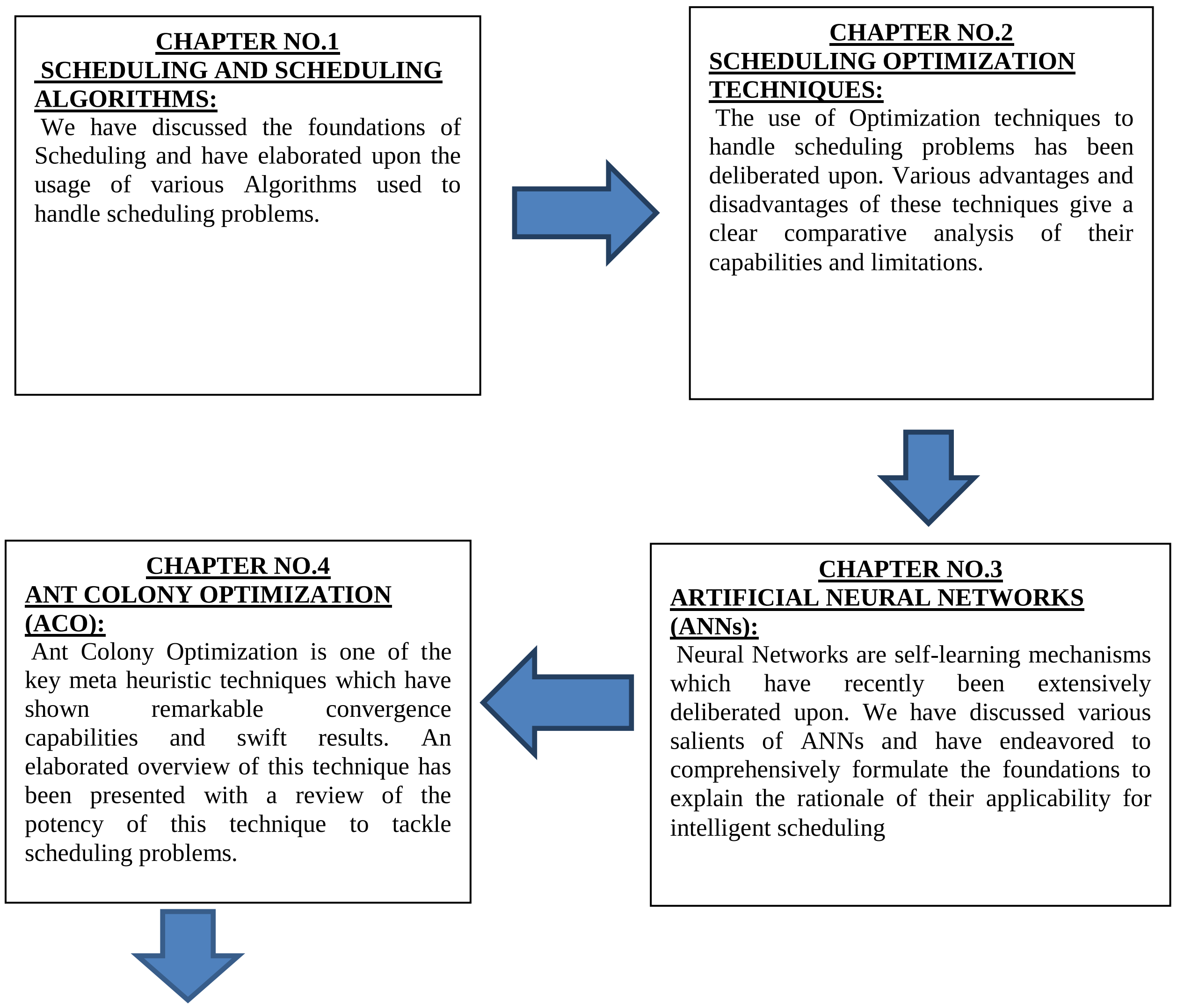
List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

This thesis is an endeavor to propose implement and analyze the augmented capability and capacity of meta-heuristic algorithms once they are combined with the self-learning mechanisms. The environment used to study these capabilities is a scheduling environment. As such I have been able to comprehensively carry out a detailed diagnosis and prognosis of amalgamating a meta heuristic i.e. Ant Colony Optimization (ACO) with Artificial Neural Networks (ANNs) to evolve a novel model of NaACO (Neural Augmented Ant Colony Optimization) to tackle scheduling problems.

The thesis is organized according to the following flow chart:



CHAPTER NO.1
SCHEDULING AND SCHEDULING ALGORITHMS:
We have discussed the foundations of Scheduling and have elaborated upon the usage of various Algorithms used to handle scheduling problems.

CHAPTER NO.2
SCHEDULING OPTIMIZATION TECHNIQUES:
The use of Optimization techniques to handle scheduling problems has been deliberated upon. Various advantages and disadvantages of these techniques give a clear comparative analysis of their capabilities and limitations.

CHAPTER NO.3
ARTIFICIAL NEURAL NETWORKS (ANNs):
Neural Networks are self-learning mechanisms which have recently been extensively deliberated upon. We have discussed various salients of ANNs and have endeavored to comprehensively formulate the foundations to explain the rationale of their applicability for intelligent scheduling

CHAPTER NO.4
ANT COLONY OPTIMIZATION (ACO):
Ant Colony Optimization is one of the key meta heuristic techniques which have shown remarkable convergence capabilities and swift results. An elaborated overview of this technique has been presented with a review of the potency of this technique to tackle scheduling problems.

CHAPTER NO.5
MATHEMATICAL FORMULATION OF THE PROBLEM; FOUNDATION FOR NaACO:

Perhaps the essence to our research is the Mathematical Formulation of our problem and how we have been able to combine the two domains i.e. ACO and ANNs to come up with our unique and novel model to tackle scheduling problems.

CHAPTER NO.6
VALIDATION OF NaACO:

The formulated model in the preceding chapters is validated by application on bench mark problems.
The results have been formulated and discussed to explain the applicability and utility of the formulated model.

CHAPTER NO.7
APPLIED CASES:

After validation the NaACo technique has been applied on a case study from the industry. The case belongs to a Hi-Tech aircraft component manufacturing workshop floor (true job shop scheduling environment). The results are discussed and the formulation is conceived as per the discussions in previous chapters.

CHAPTER NO.8
CONCLUSION AND FUTURE AREAS OF RESEARCH:

The novelty of our research and the salient achievements have been displayed in this chapter. What we have been able to extract and achieve both on academic body of knowledge front and on practical applied domain has been explicitly written down to summarize the value of our research.

APPENDICES

At the end for all practical and applied purposes problem sets along with codes of scheduling problems used for validation are attached as appendix so that the results and the findings can be correlated with the quantitative inputs. These problem sets can be very helpful for future researchers to carry out their initial workings and gain significant confidence.

IMPORTANT NOTE: At the end of each section “CHAPTER SUMMARIES” are presented to summarize the chapters and also to function as a link to the following chapter for the purpose of fluency.

1.1 The need for NaACO:

Scheduling is the art of allocating shared resources over time to mutually competing activities. In machine scheduling the machines are the resources and the jobs are the activities which are to be completed over a specified time. This situation has been able to present various search algorithms in order to minimize the processing times and hence gain efficiency. The job shop scheduling problems are most probably one of the hardest optimization problems which are NP-complete, Garey & Johson [26]. One of the earliest attempt to tackle such problems was made by utilization of branch and bound algorithms Fisher [20], and afterwards some other techniques emerged e.g. use of dynamic programming by Pinedo [44]. These methods required a very efficient use of operation research techniques and tools which were adopted to suit the locally confined problems. Mathirajan et al [4] have proposed efficient heuristic algorithms to resolve any large size real-life problems with comparatively low computational effort. In addition to these techniques, the use of Artificial Intelligence (AI) has led to many new heuristic search algorithms for scheduling problems. The main strength of these heuristic algorithms is that they immediately tend to converge on the optimal solutions, thus significantly reducing the processing time, but it certainly is very difficult to justify the quality of the solution and the ultimate capacity of these algorithms to go for an iterative process is also an aspect which is to be evaluated.

Single machine sequencing problems and their optimization through heuristic approach was the beginning of proposing a link between heuristic and optimized research. A more realistic and practical method for machine scheduling was provided by using Lagrange method. The method used Lagrangian multiplier to evolve an optimal solution and an up gradation loop was also incorporated to cater for the iterative nature of the problem.

In general the processing complexity of the scheduling problem may be formulated and summarized according to the following criteria:

1) Single stage, one processor

2) Single stage, more than a single processor

3) Multistage, flow shop

4) Multistage, job shop

The single stage, single processor and single stage multi processors scheduling problems are dealt with performing single processing on single or multiple resources. The multistage flow shop and job shop problems have an inherent complex nature as they require multiple resources, multiple allocations and hence require optimization. Moreover there convergence criteria for these problems also vary, some most common convergence criteria are as follows;

1) Minimum total tardiness
2) Minimum late/delayed jobs
3) Minimum resource utilization
4) Optimal/balance resource loading
5) Maximization of the production rate

There are two types of problems, *static* and *dynamic.* Static problems are those in which number of jobs and their times are available. Dynamic problems are the ones in which the number of jobs and the related characteristics change over time.

The artificial intelligence approach to scheduling problems suggested a better and more compact way of describing a "systems approach" towards "intelligent scheduling". The basic idea of the systems approach is to "*divide*" the problem into more realistic and manageable domains, and then "*conquer*" the problem by combining the desired results for the parts, Zhang et al [79]. The idea of the implementation of AI approach towards the solution of scheduling problems is to identify "*agents*" which are "*intelligen*t" and "*adaptive*".

The mentioned agents are the core ingredient and initializers of the AI techniques and present most appropriate solution. Such a developed "*systems approach* " may not yield "*perfect*" solutions but it does ensure that the system can be formulated and developed to cater the adjustments without the frequent intervention and thus it is capable of self-adjusting. NaACO is one such technique which is being proposed in this thesis. The development of a "*systems approach*" has led the evolution of a very popular domain of solving the scheduling problems known as "*neighborhood search methods*". Wilkerson & Irwin [6] developed one of the earliest neighborhood search methods. One of the latest additions to these methods is the Ant Colony Optimization (ACO) technique. Like genetic algorithm (GA) and many other methods, Ant Colony Optimization (ACO) is basically influenced by the wild search for food (foraging) behavior and how they converge to a food source. The extract is in the convergence capability taken from "*real"* and how these "*real"* ants can be mutated into "*artificial"*ants which further can be used as "*agents"*.

ACO has proved itself as a major competitor in terms of its performance when searching for an optimal solution. The emergence of this method is related to the evolution of theory of *stigmergy* presented by the French scientist Grasse [28]. The first design for this was proposed by Dorigo [36] and its initial implementation was on the Travelling Salesman Problem (TSP). Many variants of the basic algorithm were then introduced by Dorigo & Thomas [40] e.g. Ant System (AS), ATNS-OAP, MAX-MIN Ant System (MMAS), etc. The successful application of ACO on scheduling problems was showcased for a single machine weighted tardiness problem, similarly for flow shop scheduling problem the successful applications were also seen by Rajendra et al [59]. The constrained problems in terms of resources were done by Merkle et al [43].

From the initial advancement and the usage of these designs, it was evident that the assorted approach including ACO and other limited nature search methods gave a new area for research. Huang & Liao [73] presented a hybrid algorithm in which they combined ACO with taboo search for JSP. In another research, Niknam et al [35] have proposed a new hybrid evolutionary algorithm named HFAPSO to solve the Distribution feeder reconfiguration (DFR), which is the combination of fuzzy adaptive particle swarm optimization (FAPSO) and ant colony optimization (ACO). A software system towards development of an intelligent manufacturing system was proposed through software by Rossi & Dini [54], for solution of flexible job shop problem (FJSP).

ACO has constantly been known to outperform genetic algorithms (GA) in local search and convergence rates. The evidence comes from development of MMAS based heuristic algorithm proposed by Girish & Jawahar [46]. All in all, ACO has shown to have an efficacy towards providing us an efficient method to resolve scheduling problems and have shown that it possesses the core attributes to tackle complex, and ever evolving flexible job shop formulations.

Neural networks were traditionally used to represent a network or circuit of biological neurons. The biological neural networks are primarily used to maintain the functionality of the neuron system.

The Artificial Neural Networks (ANN) comprise interconnected artificial neurons, which are provided with a set of inputs and forget values, and they are "*trained*" to develop a relationship which then can be used to forecast or pre-empt a futuristic value, given a definite "*new*" input. This type of learning is referred as "*supervised*" learning of ANNs. Perhaps the simplest set of neural network is a feed forward neural network, in which the network information moves from the input nodes to the hidden nodes (if any) and finally to the output nodes. The feedback loop is not present in such a network. The subject loop is present in a back propagation (BP) network, which not only moves in the forward direction, but it also incorporates the feedback, backwards i.e., the inputs get continuously configured as composed to the outputs. Neural networks can perform two basic functions; they can be used to remember some information about the problem, Rumelhart & McClelland [55]. Neural networks can also be used to perform optimization and to satisfy the conditions of the given constraints by Hopfield & Tank [33]. The later form of neural network handles the job shop scheduling problems. Numerous approaches have been formulated to solve the scheduling problems through neural networks, Di Caro [27].

Two of the most popular approaches are the branch and bound methods Martin [41] and that of simulated annealing. The shifting bottleneck procedure proposed also gives sufficient evidence that neural networks can be efficiently used to solve scheduling problems. Sastri & Malave [20] have applied a Bayesian classifier in the calculation of expected cost per time period and thus determining the overall optimal control policy. Neural networks have also joined hands with ACO to put forth yet another dimension for the solution of scheduling problems.

Evidence of combined strength of ANN and ACO is evident from the work of Huawang & Wanqing [21] in which the author has used the ANN with ACO, employing the back propagation (BP) algorithm for assessing the performance of residential building. Irani & Nasimi [22] have developed a technique to use ACO with ANN for permeability estimation of a reservoir. Moreover, researchers also have discovered new neuron model, a ground for Compensatory Neural Network Architecture (CNNA), having less number of interconnection among neurons which decreases the computing time of training (Sinha et al [23]).

In this thesis, the proposed combination of ACO with ANN for tackling scheduling problems poses a novel paradigm to solve combinatorial optimization problems. In particular, the strength of ANN can be optimally utilizing the scheduling, so that the pheromone levels are obtained and updated by the use of supervised learning ANN.

In thisresearch on scheduling, optimization and meta heuristic based methods to gain efficiency has significantly been able to point out that the futuristic path to gain self-learned intelligent and optimized results is to hybridize the islands of individual automation and perfection and to combine seamlessly the concepts of intelligent solution finding algorithms with state-of-the-art exploratory and evolutionary mechanisms. The key contribution of this research is to vividly engage two distinct approaches to gain mutual advantage and to solve and present a flexible and realistic model of solution finding in scheduling.

The main contributions of this research, although explained in the relevant sections and comprehensively discussed in chapter #8, but, for the interest of the reader are as follows:

- The research has endeavored to produce a reality based solution.
- This research has been able to formulate an intelligent solution to scheduling problems.
- Formulation of an intelligent combitorial algorithm.
- Formulation of a heuristic-neural hybrid technique to tackle FJSSP.
- Development of a flexible and customized quantitative model.
- This research has used ANN for post conditioning of a Meta heuristic.
- Amalgamation of Pheromone Up gradation Variable (Tau) of ACO with ANN.
- This research has extended further the utility of swarm intelligence meta-heuristics through the amalgamation of three concepts i.e. algorithms, artificial neural networks in the application area of scheduling.
- Extra-ordinary results were obtained by ACO during this research for the formulation of NaACO.
- Novel formulation and coining of the term NaACO to be referred by future researchers.
- Novel introduction of Neu (Tau).

- Novel composition of i-ACO.
- The true nature of this finding is that through the usage of the proposed model the viability and combinations of repetitions at process and sub process levels can also be predicted and forecasted hence giving ACO a forecasting capability.

1.2 Scheduling

Researchers are focusing on scheduling and scheduling efficiencies since last four decades. Scheduling techniques are very important and have multiple usages in manufacturing, and services sector. While applying the manufacturing techniques one machine can focus only on one activity at one time. A very important topic in research of scheduling is use of complex theory with enumerative algorithm. Researchers apply enumerative algorithm to solve NP-hard scheduling problems. Researchers use empirical methods to evaluate the performance of heuristics. Sometimes it is possible to analyze the theoretical aspect and performance of heuristics. In certain case where enumerative algorithms are unsolvable solutions are generated by heuristic methods. It is quite possible that the solution might not be optimal but these solutions can be very efficient and optimized.

1.3 Scheduling Models

If a simple machine scheduling problem is considered it may be described as follows:

- m=machines to process n jobs.
- Schedule sample for each machine is i (i = 1; : : : ;m) and each job is j (j = 1; : : : ; n).
- The process involves one or more than one time interval.
- A schedule is only feasible if there is no overlapping of time intervals in accordance to some job. It simply means that a job cannot be processed by two machines at one time.

1.3.1 Machine Environment

A machine can be operated with different configuration combinations. An operation refers specifically to period of processing by machine type. In a single-stage production system, each job is completed by one operation whereas in multi-stage systems the jobs require more than one operation to production.Single-stage systems involve one machine, or m machines operating in parallel and each machine performs the same function. It is possible that machines are operating in parallel but have different speeds while they are identical. In such cases process time on a job depends on machine assignment.

1.3.2 Job Characteristics

Jobs can be classified as per ability of processing and dependency percentage of the other simultaneous jobs. In old scheduling techniques it is considered that the schedule planner has all the information relevant to the job and problems. This information contains total number of jobs planned in schedule, release dates of the jobs and how much time is required for the process of scheduling job. In classical scheduling techniques it is considered that the schedule related operations are irreversible. Experimentation proves that some scheduling problems are easy to solve while other are difficult. Computational problems can be solved step by step banking on hill climbing algorithms. Thus to summarize the basics of scheduling environments following terms are defined:

- **Routing:** The path of the operation including the work centers, time limits and the stations involved.
- **Bottleneck:** A situation in which the resources or the capabilities fall short of the required outputs or results.
- **Due date:** The time for the completion of an operation and a job sequence.
- **Slack:** The time which is included in the due date due to delays and constraints.
- **Queue:** A waiting line

1.3.3 How to Sequence Jobs

There are several techniques which are available to do short and long term planning to sequence the jobs in a scheduling environment. These techniques are based on the capacity and priorities set on the onset of scheduling operations. These rules are known as priority rules:

- o There are Decision rules which are governed by the decisions to give priority to a particular job or a work station.
- o Local Priority: These rules are applicable on the workstation level where the priorities are defined based on time constraints and related loading of that local work stations.

- Global Priority: These rules not only consider the local situation but they also consider the positions and loadings of the remaining workstations before assigning the priorities to the job which are arriving at a particular work station.

Some of the commonly used priority rules are as follows:

- FCFS: First come, first served
- LCFS: Last come, first served
- EDD: Earliest due date
- SPT: Shortest processing time
- LPT: Longest processing time
- CR: Critical ratio

1.3.4 Measuring Performance

The paradigm of performance measurement and ensuring efficiency within the scheduling domain has gained spotlight due to the constraints and resource scarcity. It is imperative now that the performance is to be quantitatively measured and the improvements be implemented at the workshop floor levels to ensure that the machines are adequately utilized. The concept of optimization of resources and efforts has been built in order to gain efficiency which is realistic. As such there are various benchmarked performance measurement standards such as:

- Job Flow Time: It measures the difference between the time job is completed and the time job was first available for processing. In essence it measures the Responsiveness of the scheduling process.
- Average Jobs in System: It measures the capacity of a process to handle the number of total jobs within a system
- Make span: It is used to measure the efficiency of a batch or lot of jobs. This measure is important to translate individual efficiency into collective results. This measures the efficiency of a lot or a workstation.
- Job Lateness: The time taken after the stipulated time allocated for the job, it measures the lack of efficiency or the situation arising before the creation of the bottlenecks in a scheduling environment.
- Job Tardiness: It measures how late the job was still completed after its due date. As such it is closely related to job lateness but its essence is to measure the due date performance.

1.4 Scheduling Algorithms

Generally, any well clarified calculated process which takes some units as ***input*** and gives out some units as ***output*** is known as an algorithm. Algorithm can also be seen as an instrument for solving a well-defined ***computational problem***. The requirement of the query defines the required input/output relation. The choice of algorithm relays on the quantity of items to be categorized, the stage to which the items are already categorized, the limitations of the inputs etc. Apart from the fact of the desired result, an incorrect algorithm can be of great help if the error rate of such algorithms be contained.

1.5 Greedy Algorithms

Algorithms may go through a series of steps for optimization problem, with a lot of calculation at each step. Many optimization problems are such that the use of algorithms on such problems is a difficult proposition. Choose what seems best at that particular moment; this is the concept of ***greedy algorithms***. This means that it chooses a local optimal solution with a hope that it would be a step to reach a globally optimal solution.

1.5.1 Elements of the greedy strategy:

This algorithm makes a series of choices as to achieve the optimal solution to a given problem. For every decision made in the algorithm, the choice that is right at that moment is picked and worked by. The process to develop a greedy algorithm goes through following steps:

- Determine the optimal substructure of the problem.
- Develop a recursive solution.
- Prove that at any stage of the recursion, one of the optimal choices is the greedy choice. Thus, it is always safe to make the greedy choice.
- Show that all but one of the sub problems induced by having made the greedy choice are empty.
- Develop a recursive algorithm that implements the greedy strategy.
- Convert the recursive algorithm to an iterative algorithm.

1.5.2 Greedy-choice property

The first and a major component is ***greedy-choice property***: which states that a locally made optimal solution can lead to a global optimal solution. If widely seen, the selection criteria for a choice to be made is considered, the choice that looks best in the current position is opted. At this point comes the difference between the two; greedy algorithms and dynamic programming, in the later the choice is made on each step considering and basing the choice on the subdivided problems, secondly they solve the problem in ascending order. While in greedy algorithms the choice is made on the current position and then subdivided problems arise. The choice made may be on the current problem but future problems are neglected in them. Hence, dynamic programming solves the problem in an ascending order while greedy algorithm solves in a descending order, progressing from one greedy problem to another.

The greedy algorithm enhances the efficiency in the choice being made in a subdivided problem. For example, in the activity selection problem, assuming that we had already sorted the activities in monotonically increasing order of finish times, we needed to examine each activity just once. It is frequently the case that by preprocessing the input or by using an appropriate data structure (often a priority queue), we can make greedy choices quickly, thus yielding an efficient algorithm.

Inspired By: Introduction to Algorithms (Thomas H. Cormen, Charles E. Leiserson, Ronald, Stein) [24], Neural Networks (Satish Kumar) [25]

1.6 Chapter Summary

This chapter focuses on:

- The introduction of "**Scheduling Environments**" and the basics of scheduling problems. Various terminologies regarding scheduling problem formulation has been explained. Moreover emphasis has been developed on how to link jobs with schedules. The concept of Job scheduling has been explained in detail.
- The techniques concerning "Job Sequencing" have been discussed. These techniques are based on but not limited to the following priority rules:

    - FCFS: First come, first served
    - LCFS: Last come, first served
    - EDD: Earliest due date
    - SPT: Shortest processing time
    - LPT: Longest processing time
    - CR: Critical ratio
    - S/RO: Slack per remaining operations

- The essence of an Algorithm in solving a scheduling problem is its ability to converge to a global solution in an efficient and sequential manner. Often the complexity of an algorithm overshadows the efficiency characteristic of a solution. Categorization is not the sole purpose for the development of algorithms. **This research is mostly about efficient scheduling algorithms and building an *Intelligent Interactive Interface***. One of the common parameter considered is of speed, i.e. how much time is taken in processing out the results.The concept of **Greedy Algorithm** is to adjust to the environment or solution space according to the developing situation or the shape of the solution. In scheduling environments the solution adapts to the changing scenarios of the problem environments. Hence the algorithms also are required to be adaptive and deceptive of the changing surroundings.

CHAPTER 2: SCHEDULING OPTIMIZATION TECHNIQUES

This research is focused on the usage of Ant Colony Optimization (ACO) meta heuristic to solve scheduling problems. An additional aspect of forecasting is implemented within the ACO through the pheromone up gradation function. Within the domain of Guided Random Search Technique this research has converged on Evolutionary Algorithms (EA) due to their mimicry of natural laws of existence and survival through which the problems have been able to find solutions from the environments in which they arise**.** This chapter briefly explains the various optimization techniques and their drawbacks to ultimately focus on EA and then towards selection of ACO as a platform for the formulation of NaACO. The real need for optimization in a scheduling environment is to reach the best realities under given constraints. The methods for optimization are distributed into three major types:

1. Calculus based Techniques
2. Enumerative Techniques
3. Guided Random Search Techniques

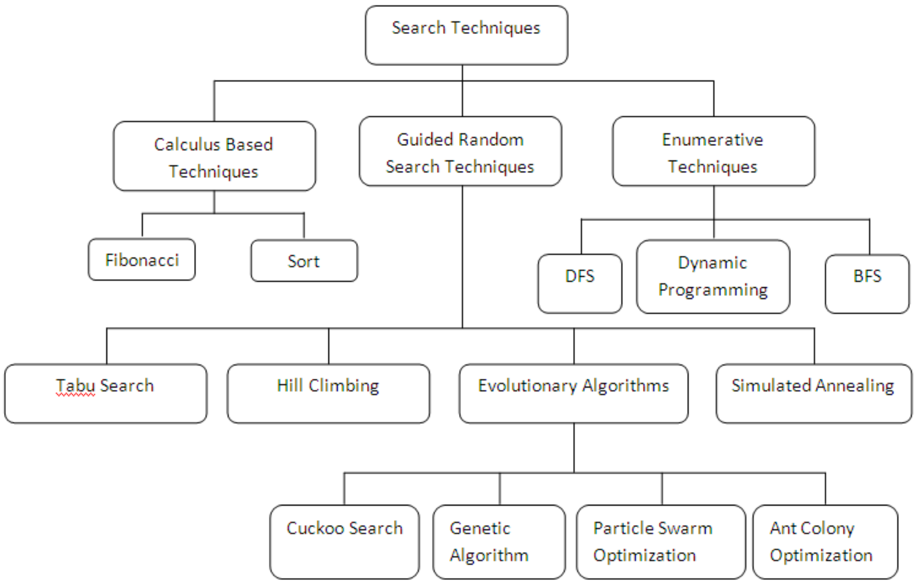


Figure 2. 1 Optimization Search Techniques

2.1 Calculus based Techniques

Calculus Based Techniques are further divided into two parts:

- **. The Direct Search Methods** finds the local maximum moving on a function over the relative local gradient directions.
- **The Indirect Methods** customarily find the local ends solving a set of non-linear equations, and then equating the gradient from the object function to zero. They involve techniques like;
    - Langrage multipliers
    - Karush-Kuhn-Tucker (KKT) methods
    - Gradient based methods
    - Conjugate Direction methods
    - Simplex Methods

2.1.1 Limitations on Calculus based Techniques

Although the calculus based techniques have their own advantages but there are some fundamental limitations associated with these techniques which are enlisted as follows:

- Provide solutions to uni-modal problems
- Convexity checks
- Regularity checks
- Implicit and coupled constraints involved
- Cannot be used for NP Hard problems
- Inefficient search of solution space
- Mathematical problems in defining real world problem with complete constraints
- Unsuitable for unconstrained problems (stuck in local optima)

2.2 Enumerative Techniques

Enumerative techniques are the methods which involve stepwise solution building and subsequent checking for the best fit solution. As compared to calculus based techniques they generally give a feasible solution but the computation time required to reach an optimum solution is reasonably higher and thus cannot be used for quick solution building.

They are mainly divided into following categories:

- Depth First Search(DFS) Technique
- Breadth First Search(BFS) Technique
- Dynamic Programming

2.2.1 **Limitations** of **Enumerative Techniques**

Enumerative techniques has been applied to a wide range of manufacturing problems which involve job shop scheduling. Within this domain e.g. dynamic programming has even provided the results within 1% to 2% of global optimum [21]. The point of concern is that the enumerative techniques work on independent resolution of the objective functions and as such the other associated objective functions of a large problem set are not getting the feedback from the previously resolved objective functions. It is inferred that this method of solution is beneficial for independent resolution of objective functions and is not effective for a problem having multiple objective functions having interdependencies towards the final resolution. All the enumerative techniques involve an abundance of computation and become more and more arduous to apply virtually in a situation involving more variables and once confronted with a huge search space.

2.3 **Guided Random Search Techniques**

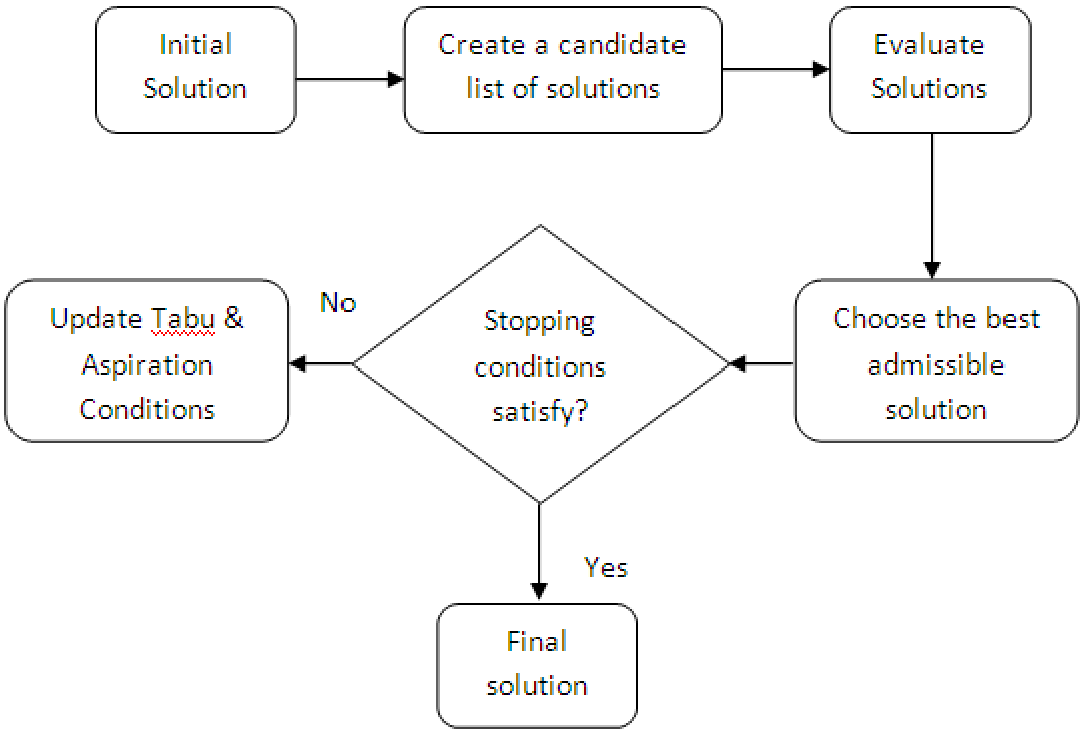
Guided random search techniques can be sub divided into following main categories;

- Tabu Search
- Simulated Annealing
- Hill Climbing
- Evolutionary Algorithms

2.3.1 **Tabu Search**

Tabu search is a metaheuristic local search algorithm which can be utilized for combinatorial optimization problems. Local search start from considering a solution to be optimum and checking its immediate neighborhood for solution finding. It is performed by having a recollection structure (which can be short, intermediate or long term) that describes the visited solution. Tabu search utilizes the exclusive recollection function to reach to final solution. The rudimentary algorithm of Tabu search by Pham et al [49] is described in figure 2.

Figure 2. 2Flow Chart for Tabu Search


Initial Solution
Create a candidate list of solutions
Evaluate Solutions
Choose the best admissible solution
Stopping conditions satisfy?
No
Update Tabu & Aspiration Conditions
Yes
Final solution

2.3.1.1 Limitations of Tabu Search

Tabu search gives initial solution to be accepted in order to evade from a local optimum and uses Tabu list for recollection function to counter repetition. It can be applied to both discrete and continuous problems. The limitation of Tabu search is that if given a considerable solution space the number of iterations required to reach an optimum solution increase considerably. Moreover Tabu search is more suited to scheduling problems in which the optimization functions and the constraints are linear as nonlinear objective functions tens to increase the solution space.

2.3.2 Simulated Annealing (SA)

Simulated annealing is a method predicated on the heat treatment process of annealing. It is a method used for finding a good approximation of global optima in a sizably large search space. It is conventionally more solution prone than enumerative methods provided if the objective is to find a good approximation of optima rather than to have an exact solution. F. Busetti [21] has given an overview on simulated annealing.

2.3.2.1 Limitations of SA

SA can deal with high nonlinear models. It's flexible and can reach to global optima. However this method is totally dependent on the standard of the proposed results and the time needed to calculate them. The accuracy of the initial population being considered can have a consequential effect on the standard of the results. Both SA and GAs, by comparison, start with an initial population, and initially concentrate on the regions of the search space found to have high fitness. This is a disadvantage if the optima is in a small region which is surrounded by all sides by regions of low fitness.

2.3.3 Hill Climbing

Hill climbing is again a local search method that moves towards the solution incrementally by having one iteration at a time. If the current solution is closer to the objective function then an increment is made to the solution until no further improvement is possible.

2.3.3.1 Limitations of Hill Climbing

Hill climbing fails to deliver an efficient solution where time is of essence as it takes incremental steps to find an optimum solution and on each step the solution has to be compared with the previous solution for fitness. This makes hill climbing an effective technique not having the requisite efficiency to converge to an optimum solution quickly.

2.4 Evolutionary Algorithms (EA)

EA is a class of evolutionary computation which uses mechanisms inspired by evolution: reproduction, mutation, amalgamation etc. The prevalent underlying concept of all these algorithms is the survival of the fittest. Evolutionary algorithms have two main distinguishing features:

- **Intensification**
- **Diversification**

Intensification enables them to find the best while diversification compels them to find solutions which can define the search space itself. The real strength of all evolutionary algorithms is that they intimate the best in nature especially the biological systems evolved in millions of years. So the forces that form the substructure of these algorithms can be summarized as:

- Variation operators that ensure the compulsory diversity
- Survival of the fittest ensures quality
- There have been many developments on evolutionary algorithms. Some of them are;
    - Ant Colony Optimization
    - Particle Swarm Optimization
    - Genetic Algorithms
    - Cuckoo Search

Venter [25] reviewed optimization techniques for non-linear constrained optimization problems through EA. Odugva et al [69] summarized the application of evolutionary computing in manufacturing industry. Since 1980s, trend of utilizing evolutionary techniques in practical applications has been on a rise. Shahram and Iraj [38] proposed a method to solve cell formation problem in cellular manufacturing by ACO. Yasuhiro Yamada et al [76] used particle swarm optimization for solving layout problems and for optimized resource allocation for a manufacturing system. Since this research focuses on evolutionary techniques of optimization, brief detail of these techniques has been discussed in the section below.

2.4.1 Particle swarm optimization (PSO)

In PSO, each particle is considered as a bird which is placed in a solution search space [27]. Each solution is composed of three dimensions, the current position, precedent best positions and the velocity. Eventually, the whole flock is liable to converge to reach the location of food i.e. global optimum position. Dr. Umarani et al [17] has discussed applications of PSO.

2.4.2 Genetic algorithms (GA)

Genetic algorithms (GA) are the most popular class of EA which are based on Darwin's theory of survival of the fittest and were first developed in 1970's by Fraser and Bernall. John Holland in 1970 first proposed the methodology of GA in detail and gave the initial framework for it. Initially a population that consists of a set of solution is considered and then evaluated on the substructure of some fitness value. Fitness value is assigned according to the objective function. A detail on elimination methods and fitness function can be found in [28]. Encoding technique varies with the problem itself. GA has been applied on scheduling problems by utilizing both phenotype and binary encoding and there are ample research papers which describe the quality of end solutions. The important aspect is the increased time taken by GA to converge on a global optima which renders this technique as an effective but inefficient met heuristic. M.K. Araffin et al [37] applied fuzzy logic to determine the optimum rate of crossover and mutation for the routing of automated guided conveyances. To understand concept of GA and crossover a simple example is given in Eiben and Smith [2]. Paris and Perrival [31] proposed that GA can be applied to deal with novel design options in manufacturing. In 2006, Omar and Baharum [39] proposed GA in solving constrained combinatorial quandary of job shop scheduling. In 2010, Y. Yang et al [74] developed a method for the dynamic facility orchestrating by utilizing GA. Apart from the manufacturing management problems of scheduling Nafis and Haque [47] proposed GA for the optimization of process for rotational components. Cylvio and Desio [36] used GA to find optimal velocity in a cruise control system. Kazem et al [5] used GA to optimize point to point trajectory for a 3-linked robotic arm. Due to the rapid magnification in this area, hybrid GAs was proposed to get more efficient results. Araffin et al [37] used fuzzy logics to control mutation and crossover rate in GA and applied it on scheduling. Likewise Kordoghli and Jmali [8] combined GA with Fuzzy logic to solve scheduling problems in a textile industry. Adnan Tariq [4] solved problem of cell formation for cellular manufacturing system utilizing hybrid GA.

2.4.3 Cuckoo search

Cuckoo Search (CS) optimization is a technique developed in 2010. It has been developed on the breeding behavior of cuckoo. There is an abundance of potential in working on this area. Cuckoo search was developed in 2010 by Yang and Deb [71]. Levy flights concept has been given to ascertain diversification along with intensification of the solution search space. Levy flight optimization methods have been discussed in [67]. Cuckoo breeding mechanism can be studied in [71] with more detail.The bottom line is that cuckoo tries to mimic the host egg continuously while the host tries to find a way to detect the parasitic egg that leads to an arms race each trying to survive out the other (survival of the fittest)

2.5 Conclusion

The fundamental objective of all of the techniques is to reach optimality,Kristina [34]. The primary concern in utlization of any of the discussed method is that they shloud not fall in local minimas. Various mathematical models have been developed for scheduling process optimization which are based on these techniques. Nourali [58] developed a mathematical model for integrated process optimization in a flexible assembly system. Ismail et al [45] considered optimization problem for scheduling processes in multiple components flow lines involving parallel assembly lines. Krishna and Rao [3] have utilized ant colony approaches while a simulated annealig approach has been utilized by Ma et al [22]. Zhang and Nee [79] have utilized a simulated annealing approach for process optimization. In all of the mentioned techniques the approach is to find a global minma with efficeincy. However, the missing link is the development of a methodology which is not only able to find optimal solution in an efficeint manner but is also able to forecast the future scenarios based on this initial finding. This research is thus focused on the development of such a technique. The main emphasis of this reseach is to focus on development of a rapidly convergent meta heuristic which is not only able to effieciently converge on an optimal solution but also should be able to predict the future scenarios based on the initial convergence. As such Ant Colony Optimization meta heuristic presents a viable platform for incorporating these features as will be discussed in the next chapter.

2.6 Chapter Summary

- The concept of optimization in scheduling environment revolves around the core notion of gaining the efficiency in convergence whilst respecting the constrained environments. The methods for optimization are distributed into three major types:
    - Calculus based Techniques
    - Enumerative Techniques
    - Guided Random Search Techniques

- Each of the above mentioned techniques have their own advantages and drawbacks. In scheduling environments combining various techniques has been known to produce efficient and effective results.
- **This research is focused on the explanation and usage of Guided Random Search techniques. Within this domain this research has converged on Evolutionary Algorithms (EA) due to their mimicry of natural laws of existence and survival through which the problems have been able to find solutions from the environments in which they arise**.

- Evolutionary algorithms have two main distinguishing features:
    - Intensification
    - Diversification
- Intensification enables EAs to find the best solution while diversification compels them to find incipient search spaces so that it can be done efficiently and the solution does not limits itself in local optima. Hence the potency of all evolutionary algorithms is that they intimate the best in nature especially the biological systems evolved in millions of years.
- So the forces that form the substructure of these algorithms can be summarized as:
    - Variation operators that engender the compulsory diversity
    - Cull acts as a force pushing quality

- There have been many techniques which have been developed within evolutionary algorithms. Some of them are:
    - Ant Colony Optimization
    - Particle Swarm Optimization
    - Genetic Algorithms
    - Cuckoo Search

- Amongst these various techniques of EA, the Ant Colony Optimization (ACO) technique has proven to be the most efficient and convergent friendly. This technique is inspired by the foraging behavior of ants and their food search pattern recognition. This technique involves the use of Pheromone Trails (a substance secreted by ants on their walking paths once they go out for food search) to converge or to continue search based on the principles of **Reinforcement** or **Evaporation**.

- This research is focused on the usage of ACO to efficiently gain an initial feasible solution and then how to combine this solution with the real time picture through the usage of Artificial Neural Networks (ANNs). This research is novel in a way that it not only addresses optimized results through ACO but it also is targeted to suggest interactive and futuristic scenarios by building an interface while keeping these techniques in the background. (The concept is continued in the next chapter)

CHAPTER 3: ANT COLONY OPTIMIZATION TO FORMULATE NaACO

This research has been inspired by the efficacy of ACO to converge and hence efficiently gain a feasible solution in scheduling problems. This chapter discussed this inspiration and how ACO has been used to provide solutions in scheduling domains.

3.1 **Preamble**

This research is novel in design as it combines a formidable convergence meta heuristic i.e. ACO with ANN in order to bring about a framework for future predictions. The capabilities of ACO to converge and that of ANN to predict are combined to form NaACO. The novelty of this combination shall prove worthy of further experimentation for future researchers.

Even though ants become annoying when they enter in households but by nature, ants are social insects; they help the environment, live in the form of group or in large nest named colonies. Population of ants in a colony depends on the species. A colony has three types of ants in it: Queen Ant, males and female workers. The queen has wings and it is the only ant that lay eggs. Males also have wings and their job is to mate with queens, subsequently they do not remain alive for long time. The other type of ants called workers does not possess wings. The number of queens a colony could have depends on the species. When queen reaches adulthood, she passes her life in laying eggs. Ant colonies possess solider ants too. The crucial tasks played by solider ants are: food collection, provide protection to queen, defend colony against enemies, and for nesting space and food source attack from colonies of enemies. If they win the battle against enemies, they take the eggs from the nest of these ants. Infant ants which emerge from these eggs spend their life as a slave in the colony. Collection of food, construction of nest, and to keep eggs and babies under great care are the few jobs of the colony. Ants are intelligent creatures. They adopt to the environments in which they dwell. According to Tom Collett and Paul Grahamin their book chapter "The Visually Guided Routes of Ants":

"*The foraging routes of ants, as displayed by columns of ants following odor trails, have long interested naturalists. Here we focus on the less spectacular visually guided routes of individual ants for what these routes can reveal about the spatial knowledge that ants have acquired of their local environment and the ways in which ants learn and use this information. The first example comes from Santschi (1913), a Swiss physician who spent most of his life practicing* *medicine in Tunisia, but whose major avocation was the study of ants (see Wehner, 1990 for a biographical sketch). The diagram shows the route followed by a single desert ant (Cataglyphis bicolor) between its nest and feeding site, as it weaves through scrub. Although, these ants forage individually and do not lay chemical trails, the details of the multiple nest bound and food bound paths shown here are intriguingly similar from one trip to the next, suggesting the importance of visio-motor memories in guiding routes. The second example is taken from a monograph by Cornetz (1910). He was a French civil engineer working in Algeria, who also spent many hours recording the trails of ants in North Africa. He walked behind them, inscribing their paths with a stick in the dust, and afterwards transcribed the tracks onto paper. This particular group of tracks displays the result of a revealing experiment. The first trace is the homeward path of an ant after it had fed. Just before the ant reached the nest, Cornetz caught the ant and carried it in his hat back to the site where it had previously been given sugar. On release, the ant repeated its previous homeward route, presumably guided by the same visual landmark memories as before*."

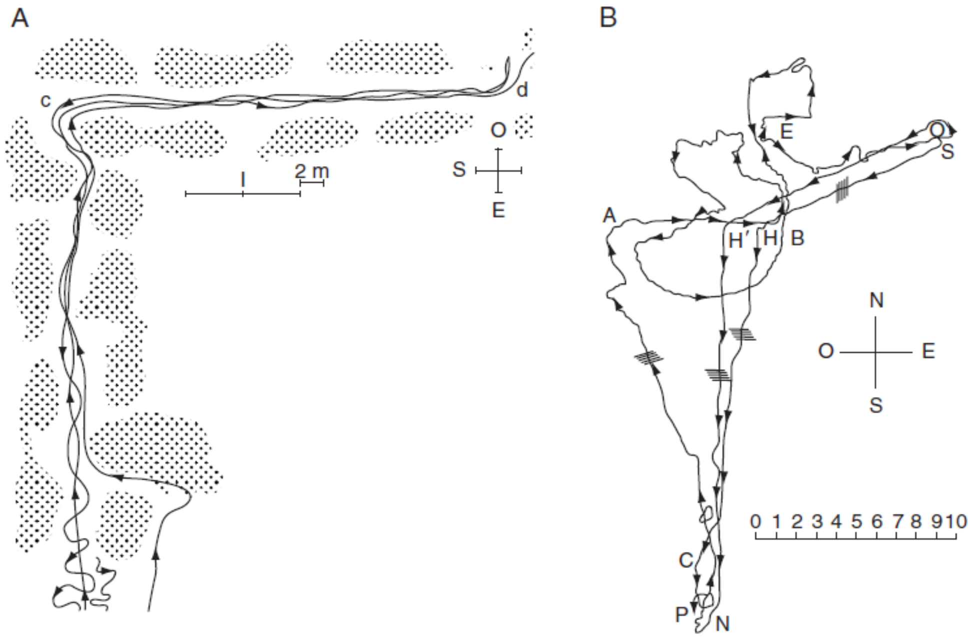


Figure 3. 1Path integration in desert ants. (Source: Book chapter "The Visually Guided Routes of Ants" by: Tom Collett and Paul Graham)

Real ants do exhaustive search for food source, once they find food they move back to their colony by following the shortest path they know. On their way back to home, ants tag their path with pheromone. Periodically, the evaporation of pheromone trial will begin to take place when other ants are not following it. Moreover, if ants find a shortcut route to the food source while following the trail then they will deposit their own pheromone trail on their routes. Eventually, the shortcut route will be picked by the follower ants. In this way, old trail’s scent will be lost by evaporation and the scent of preferred trail will become stronger. Through this strategy, ants increase their routes and search the best accurate path to the target. By doing research, scientist have succeeded to copy this strategy with their own description of pheromone trail. This “artificial pheromone trail” gives its amazing power of intelligence to ant algorithm, essential for meta-heuristics. Achievement of successful path increases the probabilities, which are the symbol for the updation of pheromone trail. Whenever a parcel reaches a place after which there are many possible paths, it chooses its path by the past experience through conveyed learning process. Regular ants in ACO carry packets from one location (origin) to another location (destination) in a well-organized way. Initially,probabilistic routing table is used by ants to determine which trip to choose to reach the destination.

In the path, nodes will behave like check points where ants will look at their probabilistic routing table. Probability of route to be taken again will be increased, only if that route is chosen by the follower ants more frequently and is verified as most efficient route. It stimulates all the regular ants to come together on prescribed path and all the packets will choose it too. The ants are declared to be stable when this phase is accomplished. These are the steps of the technique to identify the fastest path via the topology. Uniform ants are used to catch the quickest route by the structure and use more intelligence than the regular ants. Uniform ants go to seed the “probabilistic routing tables” while traveling the network which will guide the worker ants through their heuristic to take the finest path. The aim of these ants is to discover the fastest paths to various nodes, so the destination is not as much necessary for them. This is very crucial as the origin node may not be familiar with the whole layout of network. Uniform ants are also unresponsive to the probabilities set through the previous route to assist regular ants. This makes them unresponsive to the oscillation problems in the network. Backwards reporting methods is used by these ants. They give information about their status to the previous node when they reach a destination. The router updates their “probabilistic routing table” according to this information. Heuristics of these ants are used to pick the next route and to ensure that at a given node all the uniform ants are not taking the same path. Each path has equally likely chance to be taken. Ants have another trait referred as time to live which confirms that these ants will not forever crawl around the network. At every node variable passes by worker ants is increased by one. Ant dies when the value of variable hits a certain limit. The probability of an ant to choose a particular path is given by:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \text{ ; if j} \in N_i^k, \qquad \text{Equation 3.1}$$

In the above-mentioned equation, $\tau_{ij}$and η_{ij} represent the pheromone quantity and heuristic information i.e.$\eta\, i = 1 / d\, ij$ respectively. α& β are the two parameter which regulate the amount of pheromone and heuristic information .

3.2 Why ACO?

Foraging behavior of real ant colonies is the foundation of Ant colony optimization algorithm. Still much work is needed to be invested in its algorithm approaches to improve the performance. With the incorporation of local search, Ant Colony Optimization (ACO) became a competent approach to solve a variety of optimization problems. In the early days, it was proposed as an Ant System (AS), primarily based on the foraging (search for food) behavior of the ants. Ants in their colonies, operate intelligently and find food source by randomly exploring the area surrounding their nest. When ants discover source of food, they make to and fro repeated trips to carry and store food in the nest. In doing so, ants deposit a biochemical material called pheromone on their way back to nest. This pheromone trail is source of their indirect communication. The role of pheromone trail is to guide the following ants to converge on a shortest possible route between their home and source of food. Initially, the ants may follow more than one path to the food source. Over time, the shorter paths to the food source will be more frequently travelled by the ants and hence, the rate of pheromone growth is faster. This, in result will attract more ants to follow these shorter paths to the food source in their subsequent trips. Eventually, this positive reinforcement will result in the colony of ants to follow the shortest path to reach the source of food and thereby optimizing the ants' search. Ant Colony Optimization is an inspired-algorithm built on the strategy of natural ants, and was introduced to solve a number of combinatorial optimization problems (COPs) like Quadratic Assignment Problem (QAP), the famous Travelling Salesman Problem (TSP), Knapsack Problem (KP), as well as more complicated variants of these problems. Nature of the ACO algorithm makes it extremely suitable to solve the commonly combinatorial and assignment optimization problems termed as assignment type problems.

In addition, ACO has also met success in the domain of Job Shop Problem (JSP), and still it has extensively been applied in this area of research. The transposition of the ants' foraging behavior into the framework of algorithm, used to obtain solution for COPs is obtained by the correlation between:

- The food paths followed by the real ants and the appropriate solutions set for the combinatorial problem.
- The quantity of food held by a source and the function to be optimized.
- The pheromone trail and memory adaption of solution.

Ant Colony Optimization (ACO) is a community oriented, collaborative algorithm. Artificial pheromone trail is the heuristic information and guide for the ants. The pheromone trails are associated with solution components. Probability based solution are constructed, which favor those solution components which have significant heuristic information and high pheromone trail. Randomized construction heuristics implemented by ants are not similar to greedy heuristic. In randomized construction heuristic, components are added to partial solution based on probability. In general, two phases composed ACO. The first phase deals with the construction of solution, while the other phase deal with the updation of pheromone trail. In the second phase, pheromone trail is reduced by evaporation factor. This reduction is required to circumvent the limitless accumulation. Afterward, the amount of pheromone proportionate to the solutions quality is deposited by ants to highlight the component of their solutions. Generally, in ACO algorithms the most crucial fragment is to determine that how better solution for the coming cycles of algorithm get generated by pheromone trail. The main notion is to generate better solutions by merging the solution components which have been the fragment of good solutions in the previous cycles. The future cycles in ACO algorithm is influenced by past experience thereby ACO can fall under the category of adaptive sampling algorithm. Ant system (AS) is the influential work of ACO, Ant system applied in the class of NP-hard problems named TSP. For small instances, it is capable to come up with better solution, but when applied to large instances the solution quality is not satisfied. Thus, lately, many extensions to induce improvement in the performance of basic Ants system in solving TSP have been introduced. Ant Colony System, the Rank-Based Version of Ant System, Ant-Q and MAX-MIN Ant System are some of these extensions. In these extensions best found solutions are utilized more strongly, only difference is in their search control factors. In a typical manner, this is attained by performing two steps. First is in the pheromone trail, assign higher weights to better solutions. In the latter step, deposit extra pheromone trail on the global best solution arcs. Whereas, the problem of stagnation occurred when over exploitation of search experience causes ants to construct similar solutions.

The efficiency of ACO algorithm can be improved significantly by integrating local search phase, which allow few or all ants to find better solutions with the assistance of local search algorithm. Henceforth, hybrid algorithms are the advanced form of ACO algorithms. In hybrid algorithms, construction of probabilistic solution done by ants' colony is combined with the following local search phase. Hybrid algorithms construct local optimal solutions which are utilized to obtain positive feedback. In the context of broad range of optimization problems, ACO has proven itself as a competent approach to find an optimal solution. This technique is emerged and linked by the establishment of a theory termed as Stigmergy. French scientist Grasse has introduced the theory of stigmergy in 1959. This discovery addressed the co-ordination between "agents" and "environments", based on the concept of an indirect and non-centralized mechanism. The fundamental concept was that the "agents" perform actions, which become the precursor for the "following agent" to perform their subsequent actions by the use of the left traces of earlier initiators. Dorigo in 1992 proposed the first algorithm, the travelling salesman problem (TSP) was its preliminary application. Many variants of the basic algorithm were then introduced by Dorigo & Thomaslike MAX-MIN Ant System (MMAS), ATNS-OAP and Ant System (AS), etc.

3.3 **ACO and JSSP**

The job shop scheduling problems are most probably one of the hardest optimization problems which are NP-complete [1].One of the earliest attempt to tackle such problems was made by utilization of branch and bound algorithms [2]. Numerous other procedures were also settled, one of the example of these procedures is use of dynamic programming. [3]. Effective use of operation research and tools are necessary in these procedures to complement the localized nature of the problem. Mathirajan et al. [4] have proposed efficient heuristic algorithms to resolve any large size real-life problems with comparatively low computational effort. In addition, for scheduling problems number of new heuristic algorithms has been lodged by the application of artificial intelligence (AI).These heuristic algorithms has strength to immediately converge on the optimal solutions, which significantly reduce the processing time, but it is certainly very difficult to justify the quality of the solution and the ultimate capacity of these algorithms to go for an iterative process is also an aspect which is to be evaluated.

Single machine sequencing problems and their optimization through heuristic approach was the beginning of the attempts to bridge the gap of heuristic and optimization approaches. A more realistic and practical method for machine scheduling was provided by using Lagrange method. The method use Lagrangian multiplier, to evolve an optimal solution and an updation loop was also incorporated to cater for the iterative nature of the problem. In general the processing complexity of the scheduling problem may be formulated and summarized according to the following criteria

- Single stage, one processor
- Single stage, more than a single processor
- Multistage, flow shop
- Multistage, job shop

The single stage, single processor and single stage multi processors scheduling problems deals with a single processing step which should be processed on a single or a multiple resources. The multistage flow shop and job shop problems are of inherited complex nature and they require multiple resources, multiple allocations and optimization. Moreover the convergence criteria for these problems also vary, some most common convergence criteria are as follows;

- Minimum total tardiness
- Minimum late/delayed jobs
- Minimum resource utilization
- Optimal/balance resource loading
- Maximization of the production rate

Besides this, problems are considered to be statics, when the ready time and amount of jobs to be processed is available. In contrast, dynamic problems are those in which the characteristics and amount of jobs change periodically.

The JSSP was the first time conferred by Colorni in 1994[24]. The proposed Ant System (AS) algorithm has applied longest remaining processing time (LRT) as heuristic information to achieve minimized makespan. The results obtained by implementing this algorithm on ORB1, MT10, LA21and ORB4 problems were somewhat nearer to higher side. On the other hand, the shortest processing time (SPT) in AS algorithm was used by Sjoerd and Carlos [25] to minimize the makespan of JSSP. The goal of this research was to understand the influences of various setting of parameters for ACO. In another paper, Ventresca 2004 [25] has applied SPT in ACO algorithm to minimize the JSSP makespan, central goal was to explore the solution space exploration and notice the influence of pheromone update approach on it. Verification of this proposed algorithm was done by testing it on Lawrence series (LA01-LA20). The achieved outcomes were compared with Max-min Ant System (MMAS). De-Lin Lou [27] has integrated ACO with local search (ACOL) algorithm and used SPT for the role of heuristic information. This algorithm had two objectives; prime objective was to minimize the makespan and second objective was to equilibrate the workload

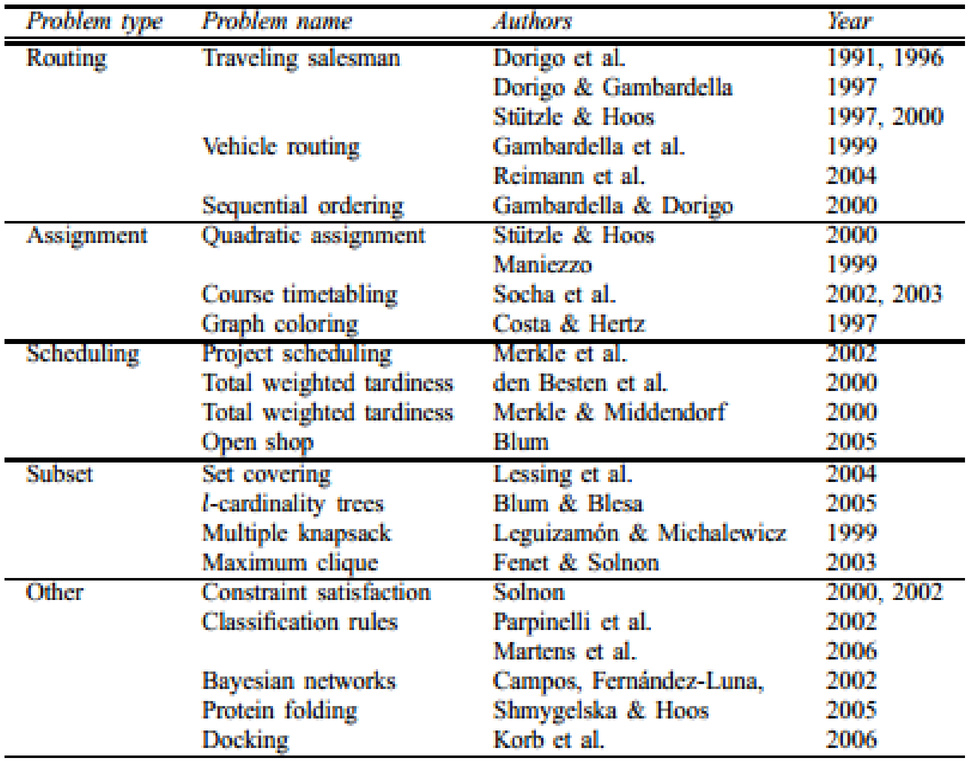
To minimize the makespan of FJSSP, Neureddine 2007 used SPT as meta-heuristic. In this paper, two algorithms ACS and ACO were used in amalgamation with Tabu Search (TS) and its performance was then compared with other existing meta-heuristic, which revealed the proposed algorithm was an effective one. Xio-Lan Zhou 2007 [28] used in Ant Colony Optimization (ACS), length of unscheduled tasks left on machine (TLM) as meta-heuristic for minimization of makespan of JSSP. The combination of simple construction with the more effective pheromone representation was used in the proposed algorithm. In a traditional manner pheromone laid on edges Ph-E, was examine with the proposed algorithm in which pheromone lay on the position of the machine required for an operation. On six bench mark problems, the proposed algorithm was tested and compared with traditional ACS. It was found that the best outcomes are yielded by proposed algorithm. To accomplish minimized makespan of JSSP, J.Heinonen and F. Pettersson 2007 [29] have employed hybrid ACO. The hybrid ACO algorithm with various visibility functions was implemented on MT10 benchmark problem. It was deduced from the analysis that better result were achieved by the combination of Length of unscheduled tasks left in job (TLJ) and Length of unscheduled tasks left on machine (TLM). The comparison between GA and proposed algorithm shows that slightly higher results were yielded by proposed algorithm than GA. Nilgun Filgah 2007 [30] used SPT for minimization of makespan of JSSP in Ant System (AS) algorithm, to fix the basic parameters of AS, author has applied the concept of Design of Experience DoE. The finest solution achieved was with higher m values and lower β values. Andrea Rossi 2007 [54] used Precedence-ordering Average Starting Time (PAST) to minimize the FJSSP makespan. On various benchmark problems, the proposed algorithm was tested. And the results came by comparing it with other meta-heuristic found in literature, declared it efficient.

For minimization of FJSSP makespan, Li Ning Xing 2009 [32] in Knowledge Based Ant Colony Optimization (KBACO) applied SPT. In this paper, comparison is done between the presented algorithm with Controlled Genetic Algorithm CGA, Temporal Decomposition, tabu search (TS) and Approach by Localization AL, AL+CGA, PSO+SA. Ultimately, competitive outcomes were attained. LiLi 2009 [33] engaged SPT in ACO to minimize the makespan, entire workload and acute machine work burden for FJSSP. This algorithm was implemented on the set of 4x5 problems along 12 operations and 8 x 8 problems along 27 operations and the other metaheuristics were matched with the last problem.. SPT heuristic was used by Ponnambalam [46] to minimize the FJSSP makespan in MMAS algorithm. The presented algorithm was executed on 13 number of benchmark problems, out of which result of seven problems have shown similar results and tagged as Best Known Solution (BKS). Among all one benchmark problem was to some extent better than BKS, whereas in five problems the result were slightly compelled to the higher side.

To minimize the makespan, average tardiness and mean flow of JSSP, Apinanthana 2010 [35] in ACO algorithm has applied least work remaining in the job (LWR), JDD heuristics and SPT. The efficiency of the presented algorithm was checked on Lawerence series (LA-01 TO LA-012), outcomes verify its strength. According to the outcomes, the presented algorithm has tendency to discover the inspiring solutions. LiLi 2010 [36] work was on the minimization of makespan, entire workload and machine work burden for FJSSP. In his research, a better algorithm was introduced; SPT heuristics was incorporated in hybrid ACA with Particle Swarm Optimization (PSO). This algorithm had received the beneficial characteristics fast convergence and positive feedback from PSO and ACA respectively. For 8x8 problems, other meta-heuristics were compared with this algorithm and the found results were slightly better.In the proposed algorithm by Rong-Hwa Huang 2012 [57], 2PH-ACO algorithm used SPT heuristic to reduce the summed tardiness and weighted earliness of FJSSP. 2PH-ACO is a strong approach used in the proposed algorithm, verified on various problems, and analyzed by comparing it with traditional ACO and Integer Programming (IP). The performance of 2PH-ACO and ACO was upgraded and robust than IP. Beside this, the problem solving ability of 2PH-ACO was better than ACO and it is applicable on problems linked to real world.

All in all, ACO has shown to have an efficacy towards providing an efficient method to resolve scheduling problems and have shown that it possesses the cone attributes to tackle complex, and ever evolving flexible job shop formulations.

Table 3. 1Application of ACO on various problem sets in scheduling domain



Problem type	*Problem name*	*Authors*	*Year*
Routing	Traveling salesman	Dorigo et al.	1991, 1996
		Dorigo & Gambardella	1997
		Stützle & Hoos	1997, 2000
	Vehicle routing	Gambardella et al.	1999
		Reimann et al.	2004
	Sequential ordering	Gambardella & Dorigo	2000
Assignment	Quadratic assignment	Stützle & Hoos	2000
		Maniezzo	1999
	Course timetabling	Socha et al.	2002, 2003
	Graph coloring	Costa & Hertz	1997
Scheduling	Project scheduling	Merkle et al.	2002
	Total weighted tardiness	den Besten et al.	2000
	Total weighted tardiness	Merkle & Middendorf	2000
	Open shop	Blum	2005
Subset	Set covering	Lessing et al.	2004
	l-cardinality trees	Blum & Blesa	2005
	Multiple knapsack	Leguizamón & Michalewicz	1999
	Maximum clique	Fenet & Solnon	2003
Other	Constraint satisfaction	Solnon	2000, 2002
	Classification rules	Parpinelli et al.	2002
		Martens et al.	2006
	Bayesian networks	Campos, Fernández-Luna,	2002
	Protein folding	Shmygelska & Hoos	2005
	Docking	Korb et al.	2006

3.4 AI and ACO

This research is about exploring the forecasting capabilities within a rapidly converging meta heuristic. As discussed in the previous section ACO has a great quality of efficiently converging to an optimal solution. This section explores the interface of ACO with AI in order to lay down the foundation for NaACO, i.e. development of an intelligent hybrid method which is equally capable of rapid convergence and of forecasting the future scenarios.

The artificial intelligence approach to scheduling problems suggested a better and more compact way of describing a “systems approach” towards “intelligent scheduling”. The basic idea of the systems approach is to “divide” the problem into more realistic and manageable domains, and then “conquer” the problem through amalgamation of achieved results for the parts [5]. The idea of the implementation of AI approach towards the solution of scheduling problems is to identify “agents” which are “intelligent” and “adaptive”. The necessary commodities and enablers for AI techniques are these “agents”, due to which it come up with effective and realistic solutions. Such a developed “systems approach ” may not yield “perfect” solutions but it does ensure that the system can be formulated and developed to cater the adjustments without the frequent intervention and thus it is capable of self-adjusting. The development of a “systems approach” has led the evolution of a very popular domain of solving the scheduling problems known as “neighborhood search methods”. Wilkerson & Irwin [6] developed one of the earliest neighborhood search methods. The concept of these methods correlates to the notion of “hill climbing.

Neural networks were traditionally used to represent a network or circuit of biological neurons. The biological neural networks are primarily used to maintain the functionality of the neuron system. The artificial neural networks (ANN) are composed of interconnected artificial neurons, which are provided with a set of inputs and forget values, and then are “trained” to develop a relationship which then can be used to forecast or pre-empt a futuristic value, given a definite “new” input. This type of learning is referred as “supervised” learning of ANNs. Perhaps the simplest set of neural network is a feed forward neural network, in which the network information moves from the input nodes to the hidden nodes (if any) and finally to the output nodes. The feedback loop is not present in such a network. Neural networks can perform two basic functions; they can be used to remember some information about the problem [18]. Neural networks can also be used to perform optimization and to satisfy the conditions of the given constraints [19, 20]. The later form of neural networks handles the job shop scheduling problems. Numerous approaches have been formulated to solve the scheduling problems through neural networks [21]. Two of the most popular approaches are the branch and bound methods and simulated annealing. The shifting bottleneck procedure proposed also gives sufficient evidence that neural networks can be efficiently used to solve scheduling problems. Sastri & Malave [22] have applied a Bayesian classifier and a BEP network in the calculation of expected cost per time period and thus determining the overall optimal control policy. Neural networks have also joined hands with ACO to put forth yet another dimension for the solution of scheduling problems.

Evidence of combined strength of ANN and ACO is evident from the work of Huawang & Wanqing [23] in which the author has used the ANN with ACO, employing the back propagation (BP) algorithm for assessing the performance of residential building. Irani & Nasimi [24] have developed a technique to use ACO with ANN for permeability estimation of a reservoir. Another research done in the field of medical diagnosis, in which researcher has trained a feed forward neural network through ant colony optimization. Moreover, researchers also have discovered new neuron model, a ground for Compensatory Neural Network Architecture (CNNA), having less number of interconnection among neurons which decreases the computing time of training [25].

The combination of ACO with ANN for tackling scheduling problems poses a novel paradigm to solve combinatorial optimization problems. In particular the strength of ANN can be optimally utilized to handle the "assigned" variables in scheduling, so that the pheromone levels are obtained and updated by the use of supervised learning ANN. This study is thus formulated to put forth such a technique, and at the same time application of this technique on various benchmark problems. The results are then formulated and future course of action is also suggested for future researchers. The following table (table 3-2) shows the extensive literature review carried out in order to ascertain and comprehend the capabilities of ACO during this research:

Table 3. 2A comprehensive survey done during this research regarding ACO and its utilization

S/No	Source	problem	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5
1	Colorni 1994	JSSP	AS		1	15	The proposed algorithm is applied on MT10, ORB1, ORB4 AND LA21 problems and results achieved are slightly on the higher side [26].
2	Sjoerd and Carlos	JSSP	AS		1	3	In this paper author has achieved the goal to gain some insight of influences of different parameters setting for ACO. Author has applied different parameter setting on many benchmark problems and achieved encouraging results [27].
3	Ventresca 2004	JSSP	ACO	Java 1.4	1	3	The main aim of this paper was to see the effects of pheromone updating technique on solution space exploration. Here a new technique called foot steeping is used which helps in space exploration. The proposed algorithm is tested on Lawrence series (LA01 - LA20) and in nine problems the results achieved are same as BKS. The results are comparable with MMAS [28].
4	De-lin Lou 2008	FJSP	ACOLS		1 and 12	(3*17)	The proposed algorithm minimizes the makespan as primary objective and balances the workload as second objective. The results achieved are then compared with ACO and AL+CGA. The results of ACOLS are better than ACO and similar

							to that of AL+CGA for makespan [29].
5	Kuo-Ling Huang 2008	JSSP	ACO	C++	1	3	The proposed algorithm has been tested on 101 benchmark problems and compared with other metaheuristics. ACOFT has yielded either equally good or slightly better results [30].
6	James Montgomry 2006	JSSP	MMAS	C language	4,5	3,16	This paper has utilized a different concept in which each machine is assigned one among a number of available alternative dispatching rules for determination of sequence for that machine [31].
7	Masaya Yoshikawa 2006	JSSP	ACO + GT Mathod		1	3	The proposed algorithm used GT method to reduce the solution space. The proposed algorithm is tested on four benchmark problems and compared with GA. The results achieved are similar as best known for these benchmark problems whereas results of GA are slightly on the higher side [32].
8	Noureddine 2007	FJSP	ACS, ACO	VB	1	3	This paper has presented two algorithms ACS and ACO with TS and the performance is compared with the results obtained from other metaheuristics and it is found that the proposed algorithm is very effective [33].
9	Xiao-Lan Zhuo 2007	JSSP	ACS		1	13	The proposed algorithm combines the simple construction and the more effective pheromone representation. Pheromone laid on edges Ph-E

							which is a traditional way is compared with (proposed) pheromone laid on the position on the requiring machine of an operation. The proposed algorithm is then tested on six benchmark problem and compared with traditional ACS. The proposed algorithm yielded better results [34].
10	J. Heinonen F. Pettersson 2007	JSSP	Hybrid ACO		1	3,4,12,13 ,(12+13)	The proposed hybrid algorithm is applied on MT10 benchmark problem using different visibility function and it is found that when TLJ and TLM are combined (TLJ + TLM) ,this weighted (70-30) visibility produces better results. The proposed algorithm is compared with the GA. The proposed algorithm produces slightly higher results as compared to GA [35].
11	Nilgu¨n Fıg˘lalı 2009	JSSP	AS	VB	1	3	The author has applied the concept of Design of Experience DoE for fixing the basic parameters of AS. The best solutions are found with higher values of m and t and with lowerβ values [36].
12	Andrea Rossi 2007	FJSP	ACS	Visual C	1	14	The proposed algorithm is tested on variety of benchmark problems and it is compared with CDMT-AS, KTS-AS, BS-MMAS, RD-GA and GT-BDDR metaheuristics found in the literature. The results achieved by the proposed algorithm are

							generally better than other metaheuristics [37].
14	Li Ning Xing 2009	FJSP	KBACO	Matlab	1	3	The proposed algorithm is compared with Temporal Decomposition, controlled genetic algorithm CGA, approach by localization AL, AL + CGA, PSO + SA and tabu search. The results achieved are competitive [38].
15	LiLi 2009	FJSP	ACA		1,8 and 9	3	This paper is applied on 4 x 5 problems with 12 operations and 8 x 8 problems with 27 operations and the last problem is compared with other metaheuristics and the results are found encouraging [39].
16	Ponnamb alam	FJSP	MMAS	C language	1	3	The proposed algorithm was applied on 13 benchmark problems. For seven out of thirteen, the results are same as best known solution (BKS). In one benchmark problem the results are slightly better than BKS and in five problems the results slightly on the higher side [40].
17	Apinanth ana 2011	JSSP	ACO	C language	1,2 and 4	3,5,16	The proposed algorithm is tested on Lawerence series (LA-01 to LA-012) to check the efficacy of the proposed algorithm. The results show that the proposed algorithm is able to find the competitive solutions [41].
18	Zhiqiang Zhang 2010	JSSP	ACO	Delphi 7	1	3	The proposed algorithm is compared with ACS and Tabu Search Algorithm TSAB for a

							number of benchmark problems and it is found that the proposed algorithm has provided better results than the conventional ACS and TSAB [42].
19	Li Li 2010	FJSP	ACA+ PSO		1,8 and 9	3	The proposed algorithm has the advantage of fast convergence of PSO and positive feedback of ACA. The algorithm is compared with other metaheuristics for 8 x 8 problems and results achieved are slightly better [43].
20	Tian Jing 2010	FJSP	ACO		1,6 and 7	3	The proposed algorithm is applied on three benchmark problems of size 10x10 and compared with the two population genetic algorithm. The results are better than the results of genetic algorithm for minimization of makespan [44].
21	Shih-Pang Tseng 2011	JSSP	HSS-ACO	C++	1	1,11	The proposed algorithm is applied on EAS, RAS, AS and ACS for testing of 10 benchmark problems taken from the literature. The results achieved improved the solution quality of ACO and its variants form 1.23% to 4.05% on average [45].
22	Rong-Hwa Huang 2013	FJSP	2PH-ACO	Lingo	11	3	The proposed algorithm presents an effective and robust approach 2PH-ACO. This approach is tested over a variety of problems and it is compared with Integer Programming (IP) and traditional ACO. Both 2PH-ACO and ACO perform better than IP. 2PH-ACO has better problem solving ability

							than ACO. 2PH-ACO algorithm can be applied to real world problems [46].

3.5 Chapter Summary

Ant Colony Optimization is a meta-heuristic technique which has been developed within the domain of Evolutionary Algorithms. This optimization technique is inspired by the colonies of ants which are in search of food sources. The ants depict a particular pattern through which they are able to converge to a food source and then the following ants are able to follow the same path through sensing a chemical known as Pheromone secreted by the former searching ants.Regular ants in ACO carry packets from one location (origin) to another location (destination) in a well-organized way. Initially uniform ant set up probabilistic routing table is used by ants to determine which trip to choose to reach the destination. In the path node will behave like check points where ants will look at their probabilistic routing table. Probability of route to be taken again will be increased, only if that route is chosen by the follower ants more frequently and is verified as most efficient route. The probability of an ant to choose a particular path is given by:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \text{ ; if j} \in N_i^k,$$

- In the above-mentioned equation, $\tau_{ij}$and η_{ij} represent the pheromone quantity and heuristic information i.e.η i $=$ 1 / d ij respectively. α& β are the two parameter which regulate the amount of pheromone and heuristic information .

- The ***ants' solution construction*** and the ***pheromone update*** are constituted by the two main phases of the AS algorithm. A good heuristic practice in AS, is to set the initializing pheromone trails to a unit marginally greater to the quantity of pheromone accumulated by ants in a single redundancy. A rough estimation of this unit can be achieved by setting $\forall(i,j), \tau_{ij} = \tau_o = m/C^{nn}$, where m represents the amount of ants andC^{nn} represents the distance of a trip formulated by the nearest-neighbor heuristic.
- **This research has enabled to establish an extensive link which supplements the application of ACO as a means of handling JSSP and FJSSP problems in the most efficient of manners. This triggering inspiration has also enabled to add an**

additional domain of forecasting through ANNs in the proposed model which was not previously catered for in classical ACO applied scheduling problems.

CHAPTER 4: ARTIFICIAL NEURAL NETWORKS FOR NaACO

Artificial neural networks form the forecasting domain for NaACO. This aspect of NaACO is discussed in this chapter as follows:

4.1 **Introduction**

NaACO is an ANN inspired formulation. In this research ACO is given forecasting capability through supervised ANN. This chapter addresses essential aspects of ANN and how is utilized in the formulation of NaACO. Essentially a typical generic ANN consists of following layers:

- **Input layer:**
  The lower most neurons are called input layer which are represented by X1 to X5 in figure 4.
- **Hidden layer:**
  The layers linking the input and the output layers are called hidden layers.
- **Output Layer:**
  The layer that gives the output or final result is called the output layer. The output neurons are represented by Z1 and Z2 in figure 4.

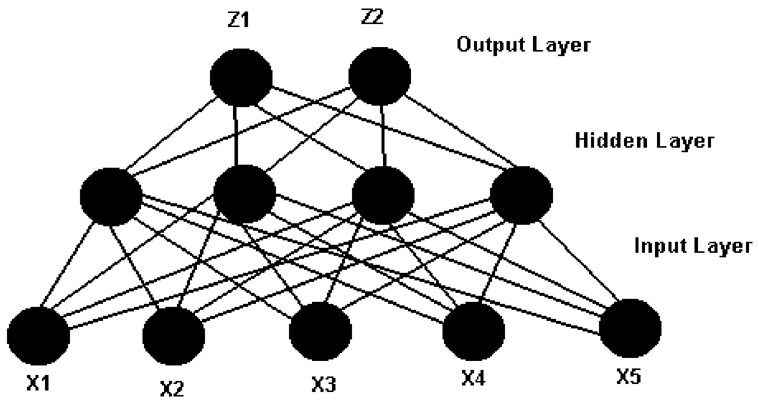


Figure 4. 1Basic Structure of Artificial Neural Network

The ANN aspect of NaACO is based upon the generic construction of a typical ANN. The input layer for NaACO constitutes the inputs from the results of ACO, and the inputs for which the forecast is required. The output layer in return gives the answer to the given query in form of a

YES or NO. The complete build-up of ANN for NaACO is presented in the following sections of this chapter.

4.2 Continuous Domain for NaACO

It is expected for NaACO that the input for which a forecast has to be made can have any number from zero to infinity, so instead of working with discrete variables, it is expedient to work in the continuous domain. Probabilities for discrete intervals of a feature measurement are then replaced by *probability density functions p(x)* such that the probability of obtaining a feature between two limits x_l and $x_u$for a particular schedule, make span or a machine processing time is given by the area under the *p(x)* between these two limits:

$$P(x_l \leq x \leq x_u) = \int_{x_l}^{x_u} p(x)dx \qquad \text{Equation 4.1}$$

This probability function is defined as Bayes' Theorem which in terms of a continuous variable is written as;

$$P(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)P(\mathcal{C}_k)}{p(x)} \qquad \text{Equation 4.2}$$

Where $p(x|\mathcal{C}_k)$ the class conditional density isfunction and $p(x)$ is the unconditional density function given by;

$$p(x) = \sum_{k=1}^{C} p(x|\mathcal{C}_k)\, P(\mathcal{C}_k) \qquad \text{Equation 4.3}$$

Where the summation is over all *C* classes. As before, the posterior probabilities must be summed to one for consistency.

4.3 Implementing Classification Decisions for NaACO

Bayes' theorem is applied with neural networks to come up with the probability distribution over the network weights, w, given the training data. This is used to take a decision on the classification of a feature vector *X*, where *X* is assumed to be a vector of feature measurements which in this research are the machine processing times, make-spans etc. The logical and straightforward classification rule is one that assigns an observed feature vector *X* to a class $\mathcal{C}_k$ that has the largest posterior probability. In other words, assign *X* to class $\mathcal{C}_k$ such that

$$P(\mathcal{C}_k|X) = \max_j\{P(\mathcal{C}_j|X)\} \qquad \text{Equation 4.4}$$

This essentially translates to saying: assign *X* to class $\mathcal{C}_k$ such that

$p(X|\mathcal{C}_k)P(\mathcal{C}_k) > p\big(X\big|\mathcal{C}_j\big)P\big(\mathcal{C}_j\big)\forall j \neq k$ Equation 4.5

Where $P(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)P(\mathcal{C}_k)}{p(x)}$ is employed in feature vector form by replacing the single feature variable x by the feature vector X and where the resulting common factor $p(X)$ in the denominator of the posterior probability cancels out.

4.4 Placement of a Decision Boundary for NaACO.

The imperative aspect in the design of a pattern classifier for NaACO is the placement of a decision boundary which separates the classes in question. Where should decision region boundaries be placed such that the probability of misclassification is minimized? To answer this question it is instructive to quantify the probability of an error occurring in the classification process. Reverting to the single dimension two class case, the decision boundary partitions the input space into two regions R_1andR_2. Then the probability $P_{error}$of a feature x being assigned to the wrong class is given by this equation:

$P_{error} = P(x \text{ is in } R_2 \text{but is assigned to } \mathcal{C}_1) + P(x \text{ is in } R_1 \text{but is assigned to } \mathcal{C}_2)$ Equation 4.6

This can be quantified as follows:

$$P_{error} = P(x \in R_2, \mathcal{C}_1) + P(x \in R_1, \mathcal{C}_2)$$
$$= P(x \in R_2|\mathcal{C}_1)P(\mathcal{C}_1) + P(x \in R_1|\mathcal{C}_2)P(\mathcal{C}_2)$$
$$= P(x > x_d|\mathcal{C}_1)P(\mathcal{C}_1) + P(x < x_d|\mathcal{C}_2)P(\mathcal{C}_2)$$
$$= \int_{-\infty}^{x_d} p(x|\mathcal{C}_2)P(\mathcal{C}_2)dx + \int_{x_d}^{\infty} p(x|\mathcal{C}_1)P(\mathcal{C}_1)dx$$ Equation 4.7

In order to minimize P_{error} the decision boundary for R_1 and R_2 is chosen such that;

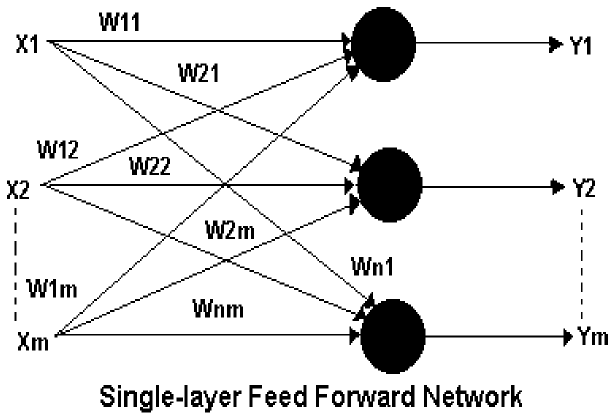
- Point x lies in R_1 (class$\mathcal{C}_1$) if $p(x|\mathcal{C}_1)P(\mathcal{C}_1) > p(x|\mathcal{C}_2)P(\mathcal{C}_2)$.
- Point x lies in R_2 (class$\mathcal{C}_2$) if $p(x|\mathcal{C}_1)P(\mathcal{C}_1) < p(x|\mathcal{C}_2)P(\mathcal{C}_2)$.

4.5 Network architectures for NaACO

For NaACO this research has considered two types of network architectures:

4.5.1 Single layer feed forward networks:

The neurons are systemized in layers in this type of structure. In this basic form of a layered network there is one way traffic i.e. input layers from where origin node is passed on to the output layer. This network can also be called feed forward network as well as acyclic structure.



The following figure explains the phenomenon discussed:

Such network is referred to as the single layer structure in which the output nodes are called "*single layer*".

4.5.2 Multilayer feed forward networks:

Another type of acyclic structure differentiates itself by having uni or multi hidden layers, whose calculation nodes are accordingly called neurons or units. Job of hidden neurons is interring venue between the extrinsic input and the structure output in a useful manner. The main function of the hidden layers is to take out repetitive values from the dense input given. The input vectors are fed forward to the hidden layers in between till the ending layer which is called the output layer, gives genuine structure replication.

Figure 4. 2Single-layer Feed Forward Network

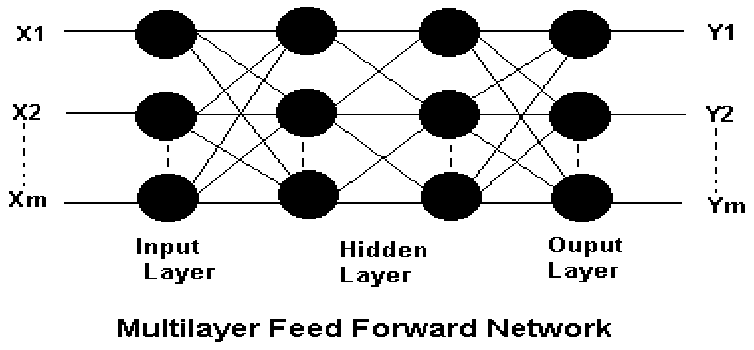


Figure 4. 3Multilayer Feed Forward Network

For NaACO as the inputs can have multiple values, the Multilayer feed forward network is chosen so that the inputs in form of machine times, job schedules and makespan can be accommodated to give forecasted outputs after the network has been sufficiently trained.

4.6 Learning of ANN for NaACO

The major characteristic of neural network is that it learns from the given atmosphere and so can enhance its execution accordingly. A neural structure knows its atmosphere by continuous interplay way through adaption implemented to its synaptic weights and inequitableness levels. The structure learns more and more with its atmosphere by more interaction to it. Generally there are two types of learning atmospheres available:

4.6.1 Supervised learning:

In this method the neural structure knows about the atmosphere it is working in by the illustrations provided by the teacher. The following figure (4.4) explains the process:

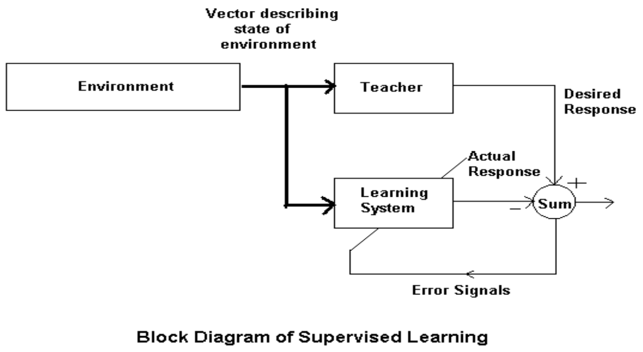
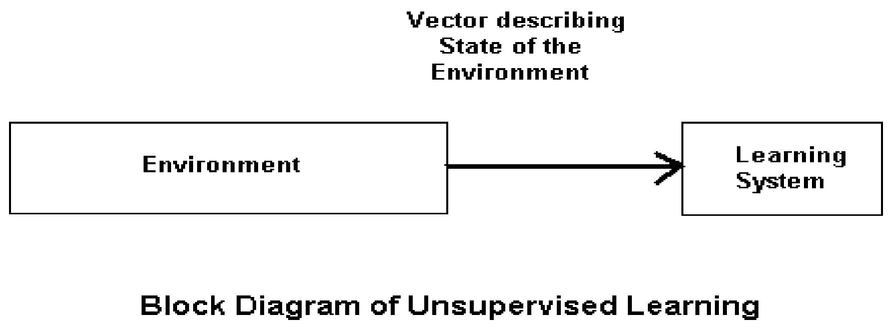


Figure 4. 4Block Diagram of Supervised Learning

Neural structure replication to inputs is in contrast to the already defined outcome or output. The margin between input and output value is then send back to the inputs along with the new inputs to get closest to the desired output.

4.6.2 Unsupervised learning:

In this process no external teacher is available to inspect the ongoing process .As shown in figure



4.6 below.

Figure 4. 5Block Diagram of Unsupervised Learning

For the purpose of this research the supervised mode of learning is selected in which the learning tasks are performed through inputs and the requisite outputs for the NaACO to prepare itself for the forecasting process. However, it is also mentioned here that the unsupervised learning environment can be further developed for NaACO and is presented as a future area of research.

4.7 Learning tasks for NaACO

4.7.1 Pattern recognition:

ANNs carries out basic cognitive process by initially going through a coaching lesson, throughout that the structure constantly gives a group of input pattern alongside the class to which every specific pattern belongs to. After going through with this iterative process the ANN is capable of classifying each input to a unique pattern and giving an adequate answer. For NaACO the classifying pattern is being generated by giving the inputs in form of machine processing times, the value of tau and the outputs are assigned based on the achieved makespan through ACO. Once the input and output data sets have been generated, the ANN is made to learn and recognize the pattern within this domain. At the end of pattern recognition the input scenarios are changed to cater for inputs based on on-ground situations for the NaACO to return the forecasted answer. The structure of NaACO recognizes the pattern provided to it. Following features are obtained through supervised learning and pattern recognition through NaACO:

4.7.2 Control

The NaACO structure is taught to learn how to control the inputs versus the outputs. The training aspect of NaACO refers to the training sets in which the limits of inputs are recognized by repeatedly giving those training sets which are defining the solutions outside the limits of the problem and associating

4.7.3 Adaptation

The atmosphere in which the NaACO structure is trained is not constant, the stats criterion of the facts that are given by the structure change with time. In such cases, the common ways of assisted learning may not proof to be of any help as they cannot cater the statistical changes that will come up in the particular atmosphere. To cater this problem, the neural structures should vary itself according to the change in the environment. This changing of the structure according to the atmosphere is called adaption of the network. This working of a structure is said to be adaptive in nature as it treats each input as a unique one and the learning process of the structure runs along the tasks given to it.

4.7.4 Generalization

NaACO is continuously fed with multiple scenarios so that the structure designed is smoothly generalized. A structure is declared generalized when the input provided and the output given by the network is accurate or a new solution comes up which is different from the illustrations provided to it.

4.8 Limitations of NaACO as regards to the Neural Interface

In general the limitations of NaACO with respect to the neural aspect are the same as that of the ANNs. These limitations also present the future opportunities for researchers to strengthen the neural aspect of NaACO.
The salient limitations are briefly stated below:

- **Low Learning Rate:**
  For problems requiring a sizably voluminous and complex specification or having large variety of training examples, the network shall require more time to get trained.

- **Imprecision:**
  ANNs do not give accurate mathematical solutions they give the most nearest output state.

- **Black box approach:**
  ANNs can change provided input to output but does not tell the methodology being used.

- **Limited Flexibility:**
  This structure can be used for single system only. For a new system the ANN has to learn the patterns again in order to forecast the future scenarios.

4.9 Chapter Summary

- The advancements in Artificial Intelligence (AI) have enabled researchers to conquest and shatter the boundaries of discrete reasoning. Instead run time or continuous solution building through intelligently analyzing and learning from the environment is taken as a new paradigm of optimized solution building. In this respect the concept of Artificial Neural Networks (ANNs) has emerged. This concept is based upon the ability of natural neurons to learn from repeated exposure to stimuli and then use the results of these past experiences to come up with a logical pattern through which they can predict future scenarios.

- The ANNs learn in Supervised Environment or in an Unsupervised/ Reinforced Environment. The former encompasses the use of Inputs and the corresponding Outputs through which a pattern is recognized between these variables and predictions are made;

whereas in Unsupervised Learning only the Inputs are used to describe a vector which represents the Input Environment. Based on this vector the outputs are forecasted and predicted. The comparative flows of these two techniques is represented as follows:

- The ability of ANNs to recognize the patterns, link and build relationships amongst likely unrelated entities through supervised or unsupervised learnings have inspired this research to develop a neural interface in the Pheromone Up gradation domain of ACO (explained in next chapters) and then to use this interface to evolve a novel model which can not only be used in solving the scheduling problems but also it can be utilized to predict future scenarios/outputs.

- This research used a Multilayered Feed Forward ANN to make it an integral part of the basic ACO technique for evolution of the proposed ACO/Neural inspired model. This model has been able to combine the fast convergence characteristics of ACO with the pattern learning capability of ANN.

CHAPTER 5: PROBLEM FORMULATION

5.1 Introduction:

In this research a novel technique has been formulated which not only focuses on fast convergence and hence the efficiency but also this technique can help in determining the future scenarios of scheduling so it is capable of forecasting. Often in industrial and scheduling situations the focus is on efficient use of resources which has inspired this research to the development of a technique which is close to the ground realities of workshop floors and manufacturing environments. ACO has been used to converge and ANNs in combination with ACO is used to build up futuristic scenarios. The complete flow of the formulation can be

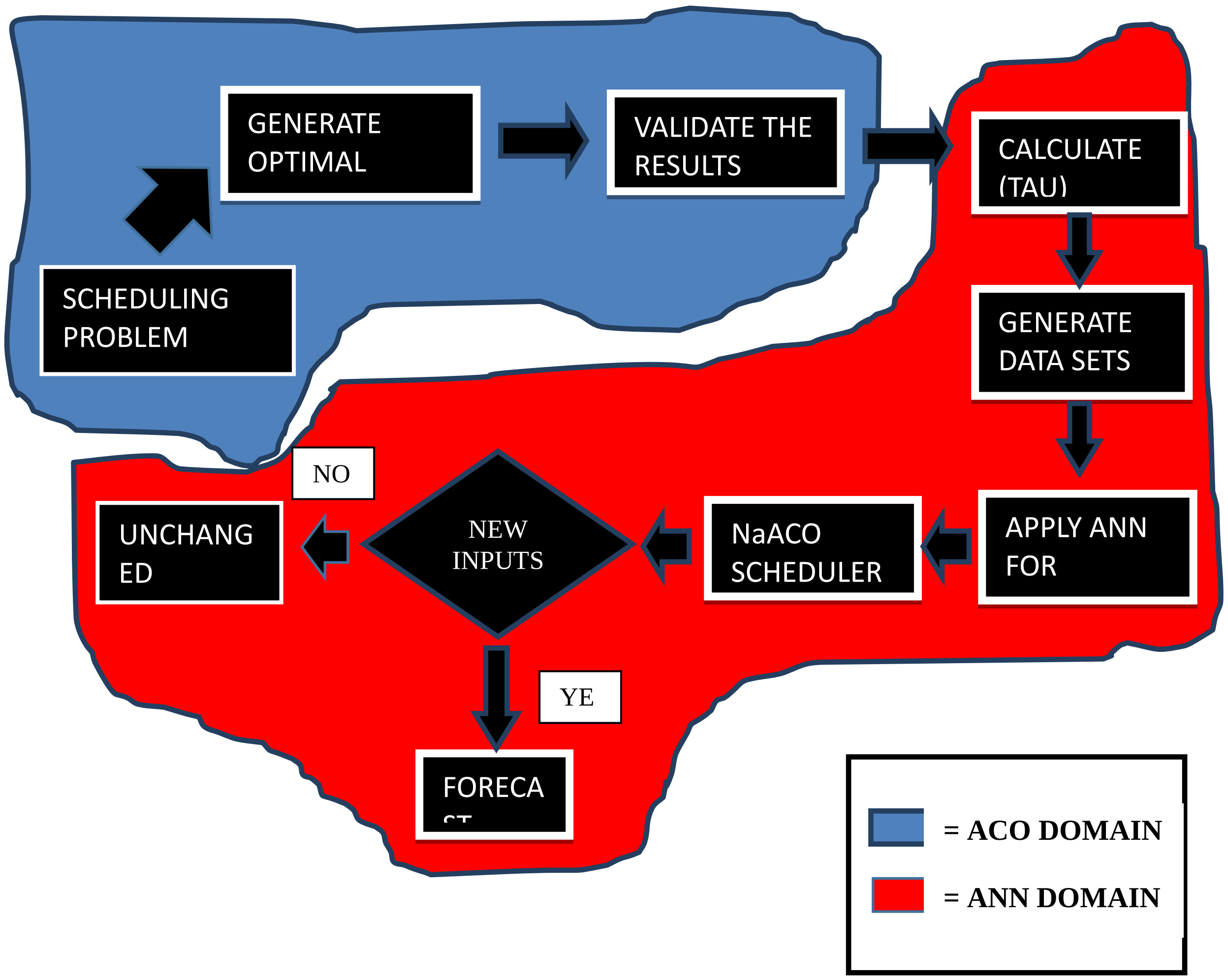


Figure 5. 1The formulation flow of the NaACO technique

summarized as per figure 5.1 below:

The detailed formulation is divided into two main parts i.e. the ACO and the ANN domain. Throughout these two parts a problem set of PC Hu [66] comprising of 100 problems (given in Appendix A) has been utilized to ascertain the correctness of the formulation. Before converging on the mathematical formulation the scheduling and optimization aspects of the research are discussed in section 5.2 and 5.3 below.

5.2 **Scheduling Aspect of the Problem:**

Scheduling on workshop floor involves the optimal use of resources and adhering to the constraints of the situation. Typically in scheduling the emphasis is laid upon the efficiency of the process and ways and means to eliminate the waste. In static scheduling situations the machines and the work centers are static and the jobs are assigned to these machines. Whereas in dynamic scheduling environments, job hopping in conjunction with non-idle workstation identification is carried out to come up with the most efficient flow of the job. In addition to these conditions, in flexible job shop problem (FJSSP) the job can be given to any machine from a selected group of machines and it is processed such that the maximum completion time for the job (make span) of all the operations is reduced to minimum.

As mentioned in *chapter.no.1* jobs can be classified as per ability of processing and dependency percentage of the other simultaneous jobs. In scheduling techniques it is considered that the schedule planner has all the information relevant to the job and problems. These information contains total number of jobs planned in schedule, release dates of the jobs and how much time is required for process of scheduling job. These problems are often combined with the formation of bottlenecks in the process where there is a single machine or a workstation which is causing all the delays. Thus to summarize the basic of scheduling domain this research is focused upon:

- **Routing:** The path of the operation including the work centers, time limits and the stations involved.
- **Due date:** The time for the completion of an operation and a job sequence.
- **Slack:** The time which is included in the due date due to delays and constraints.

The paradigm of performance measurement and ensuring efficiency (due date, slack) within the domain of this research has been linked to the question that how performance be quantitatively measured and the improvements be implemented at the workshop floor levels to ensure that the machines are adequately utilized. The concept of optimization of resources and efforts have been built in order to gain efficiency which is realistic. As such the scheduling domain of this research explicitly addresses various benchmarked performance measurement standards such as:

- **Job Flow Time**: It measures the difference between the time job is completed and the time job was first available for processing. In essence it measures the Responsiveness of the scheduling process.

- **Average Jobs in System**: It measures the capacity of a process to handle the number of total jobs within a system

- **Make span**: It is used to measure the efficiency of a batch or lot of jobs. This measure is important to translate individual efficiency into collective results. This measures the efficiency of a lot or a workstation.

- **Job Lateness**: The time taken after the stipulated time allocated for the job, it measures the lack of efficiency or the situation arising before the creation of the bottlenecks in a scheduling environment.

The following core issues are addressed in the typical parallel machine scheduling problem:

- Assigning out jobs to the machines
- Aligning jobs for each machine
- How to schedule jobs to the machines?
- How to assign workers to the machines?

For NaACO when the job assignment to the machines is done, the initial convergence of the job to particular machine develops the basis of its minimum time span to complete the job. In this research an approach has been formulated in which the initial convergence is handled by the most efficient of algorithms and meta-heuristic technique available and then a method is introduced through which the system is able to predict and determine the futuristic values based on the ground realities and the inputs achieved in the process of initial convergence. For the first part i.e. initial convergence ACO has been used and for the second part Artificial Neural Network has been incorporated and is interfaced with ACO to formulate NaACO (Neural Augmented ACO).

Continuing the problem formulation for NaACO a typical situation is presented in which there are n number of workstations/machines/desks. There are m number of jobs/workers etc. which are to be sent to those n workstations/machines/desks. Then there are some assumptions which are kept to formalize the problem set. Each workstation has some in built variables. These variables can be time, capacity, worker skill level, machine limitation etc. In essence these variables are considered to be common for all the workstations/machines/desks (a typical framework for service lines, manufacturing setups etc.). Now the assumptions/constraints developed for the scheduling domain can be explained in the following manner:

As per the scheduling formulation of the problem the research focuses on scheduling n jobs to m parallel machines and then based on the best attainable solution through ACO switch to ANN training of the results achieved to formulate NaACO which is able to forecast and predict future situations. The general assumptions pertinent to this formulation are:

- Each job has only one operation and once the job is assigned to any machine, it is with the machine till that operation is complete. It also means that the initial assignment of the jobs to any machine will render that machine to complete the job and then move to another machine.
- The machines have a range of processing times (or other variables) for each job. These processing times have different values and represent the variation which each machine can show for a particular job.
- No job pre-empts/ splitting is allowed and hence we are considering a situation in which we are not allowing any job hopping or incomplete job retrievals.
- Any machine can process any job first. This parameter gives us the flexibility to assign any job on any machine once we are about to start. For this purpose we require a triggering heuristic function which can place the initial job at the right machine. This

triggering heuristic can be the machine processing times, the requirement of the process i.e. the precedence already set by the process limitations etc.

- Machine setup times are negligible. Although in reality the machines do have setup times and these setup times cause delays in the overall operations, but for our purpose we have catered for these delays in the machine processing times as indicated in the above constraints.

- Transportation time between the machines is negligible and the transportation time is taken within the operation time of the machines

- Number of jobs and machines are fixed.

- All m machines, n jobs are available at time zero.

- This Overall processing time for the whole setup is the sum of individual process times. Now ideally speaking this time has to be minimum to cater for the requirements of efficiency and productivity in any setup. Realistically speaking there are many other variables which have direct or indirect effects on this total processing time. The effect of all of these variables are translated into the processing times of each workstation/machine desk.

5.3 The Optimization paradigm of the Problem:

As mentioned in *chapter.No.2* the concept of optimization in scheduling environment revolves around the core notion of gaining the efficiency in convergence whilst respecting the constrained environments. This research is focused on the explanation and usage of Guided Random Search techniques. Within this domain the research has converged on Evolutionary Algorithms (EA) due to their mimicry of natural laws of existence and survival through which the problems have been able to find solutions from the environments in which they arise. The details regarding the justification of this domain for scheduling problems has already been explained in *chapter.no.2*. Amongst these various techniques of EA, the Ant Colony Optimization (ACO) technique has proven to be the most efficient and convergent friendly. This technique is inspired by the foraging behavior of ants and their food search pattern recognition. This technique involves the use of Pheromone Trails (a substance secreted by ants on their walking paths once they go out for food search) to converge or to continue search based on the principles of Reinforcement or Evaporation. The optimization paradigm of this research is hence focused on usage of ACO to efficiently gain an initial feasible solution and then to combine this solution with the real time picture through the usage of Artificial Neural Networks (ANNs). This research is novel in a way that it not only addresses optimized results through ACO but it also is targeted to suggest interactive and futuristic scenarios by building an interface while keeping these techniques in the background. With reference to the elaborated catharsis of ACO and its application in scheduling in *chapter.no.3*, at this stage it is reiterated that this research has been able to establish an extensive link which supplements the application of ACO as a means of handling JSSP and FJSSP problems in the most efficient of manners.

The mathematical formulation of NaACO in the context of section 5.2 and 5.3 is presented in the following parts of this chapter.

5.4 Mathematical Formulation for NaACO

The mathematical formulation is divided into two parts. The first part (part-I) concentrates on formulation for ACO domain and the second part (part-II) focuses on formulation for ANN domain (as depicted in figure 5.1). The complete formulation is depicted in the following figure

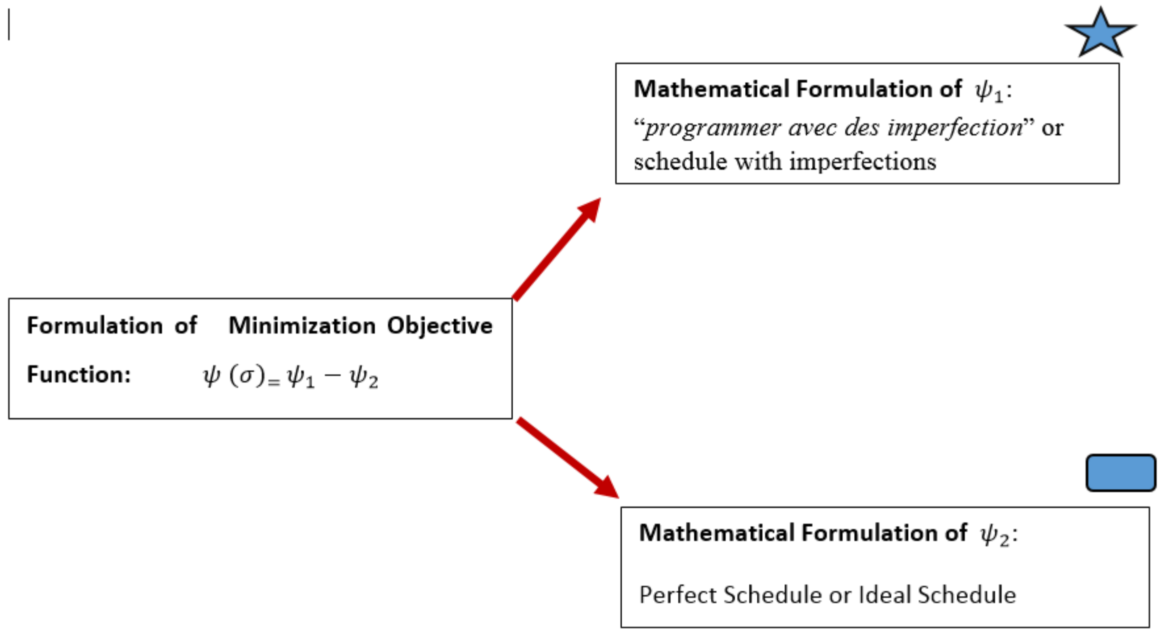

Mathematical Formulation of ψ_1:
"*programmer avec des imperfection*" or schedule with imperfections
Formulation of Minimization Objective Function: $\psi(\sigma) = \psi_1 - \psi_2$
Mathematical Formulation of ψ_2:
Perfect Schedule or Ideal Schedule

Figure 5. 2Mathematical formulation flow for NaACO

5.3:

As depicted in figure 5.3 the formulation revolves around formation of a minimizing objective function $\psi\,(\sigma)$ comprising of the outputs from both the parts of NaACO i.e. the ACO component ($\boldsymbol{\psi_2}$) and the ANN component ($\boldsymbol{\psi_1}$). This objective function reflects the deviation between the realistic solution (i.e. the forecasted value attained through ANN component i.e.$\boldsymbol{\psi_1}$) and the ideal solution (i.e. the ideal value attained through the initial application of ACO i.e. $\boldsymbol{\psi_2}$). If there is no deviation then the objective function returns a zero value ($\boldsymbol{\psi_1 - \psi_2 = 0}$). This is the minimum value of this objective function as the realistic, on ground solution cannot be better than the ideal solution as it will always have some inherent delays, slacks and impediments.

5.4.1 Part-I: ACO Domain of Formulation (Formulation of$\boldsymbol{\psi_2}$)

After the evidence required to ascertain the convergence capabilities of ACO and describing the scheduling domain of the problem, the formulation now converges on the application of ACO. The main concern of scheduling in the widest sense is to allocate resources to tasks over time. The formulation is explained as under:

5.4.1.1Notations of the Problem

The research focuses on scheduling$n$ jobs to m parallel machines as per the following constraints:

- Each job has only one operation.
- The machines have range of processing times (or other variables) for each job.
- No job pre-empts/ splitting is allowed;
- Each job has its own due date;
- Any machine can process any job first.

- Machine setup times are negligible
- No machine may process more than one job at a time
- Transportation time between the machines is negligible.
- Number of jobs and machines are fixed
- All m machines, n jobs are available at time zero.

5.4.1.2Proposed Structure of the ACO Algorithm

The jobs are scheduled as per the application of ACO algorithm. The probability of any job (ant as per ACO) to go to any machine (food source as per ACO) is calculated by the following equation:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \text{ ; if j} \in N_i^k \qquad \text{Equation 5.1}$$

In this equation the $\eta_{ij}$is the reciprocal of the heuristic function. In this formulation the heuristic is the processing time taken by each machine.$\tau_{ij}$is the pheromone updation function which is used for reinforcement or evaporation. Also α and β are the sensitivity factors for η_{ij} and τ_{ij} which vary with each problem set being evaluated. The sequence of generation of an algorithm for the above mentioned equation (5.1) is as follows:

- Initialize heuristic parameters. Set n as the set of schedulable jobs for m machines.
- While (termination condition is not met) do
- For each ant in the colony do
- Apply local search.
- For (n=1, 2, 3… n) do
- Identify all schedulable jobs and include them in the partial solution.
- Apply local updating rule to the pheromone value compatible with the solution being constructed.
- **End for**

- Apply global updating rule to the best solution found so far or the iteration best solution.
- Apply local search algorithm to the iteration best solution and in case of improvement, update the best solution found so far.

As indicated earlier to ascertain the correctness of the formulation a set of 100 problems of Pc Hu have been utilized. The problem itself comprises of scheduling 12 jobs on 3 machines. These problem sets were jointly tackled by the researcher at the start of this research to examine the effectiveness of ACO. Each machine has three variable times represented by A, B and E. A sample problem set is represented in table 5.1 below:

Table 5. 1Sample problem set used during the formulation (time in secs)

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	3	596	6	6	587	8	1	767	2
2	2	70	9	7	221	1	1	433	9
3	1	507	6	1	285	5	0	395	8
4	5	570	4	8	346	7	7	778	7
5	2	12	7	2	755	8	5	174	4
6	5	344	1	9	97	3	5	303	1
7	3	321	7	0	516	9	2	316	2
8	8	220	7	0	278	8	4	225	2
9	4	788	5	7	83	7	7	402	5
10	1	642	2	3	148	9	7	110	4
11	6	556	3	4	62	8	3	413	1
12	7	334	3	1	346	4	9	772	3

As a first step the heuristic parameters are initialized by calculating the η_{ij} values. For these problem sets the total processing time taken by each machine is given by A+ B/E. So for the first row in front of job.no.1 in table 5.1 the total processing times taken by Machine1, Machine2 and

Machine 3 are given as 102.3, 79.375and 384.5seconds respectively. These times are reciprocated to generate the triggering heuristic for initialization of ACO. The reciprocal values of the timings is represented in table 5.2 below. These values refer to the formation of η_{ij} through which the probability for an ant to converge to a particular food source shall be calculated.

Table 5. 2Reciprocal of the heuristic function for the initialization of the first step.

Job	Machine 1				η	Machine 2				η	Machine 3				η
	A_i	B_i	E_i	A+B/E	1/T	A_i	B_i	E_i	A+B/E	1/T	A_i	B_i	E_i	A+B/E	1/T
1	3	596	6	102.3333333	0.009771987	6	587	8	79.375	0.012598425	1	767	2	384.5	0.00260078
2	2	70	9	9.777777778	0.102272727	7	221	1	228	0.004385965	1	433	9	49.11111111	0.020361991
3	1	507	6	85.5	0.011695906	1	285	5	58	0.017241379	0	395	8	49.375	0.020253165
4	5	570	4	147.5	0.006779661	8	346	7	57.428571	0.017412935	7	778	7	118.1428571	0.008464329
5	2	12	7	3.714285714	0.269230769	2	755	8	96.375	0.010376135	5	174	4	48.5	0.020618557
6	5	344	1	349	0.00286533	9	97	3	41.333333	0.024193548	5	303	1	308	0.003246753
7	3	321	7	48.85714286	0.020467836	0	516	9	57.333333	0.01744186	2	316	2	160	0.00625
8	8	220	7	39.42857143	0.025362319	0	278	8	34.75	0.028776978	4	225	2	116.5	0.008583691
9	4	788	5	161.6	0.006188119	7	83	7	18.857143	0.053030303	7	402	5	87.4	0.011441648
10	1	642	2	322	0.00310559	3	148	9	19.444444	0.051428571	7	110	4	34.5	0.028985507
11	6	556	3	191.3333333	0.005226481	4	62	8	11.75	0.085106383	3	413	1	416	0.002403846
12	7	334	3	118.3333333	0.008450704	1	346	4	87.5	0.011428571	9	772	3	266.3333333	0.003754693

After the calculation of η_{ij} the next step is the calculations of the respective probabilities for ants (jobs) to converge to a food source (machines). For example the probability for job 1 in the mentioned problem set to go to any of the three machines is calculated as under:

$$Prob\ (Job1\ for\ Machine\ 1) = \frac{\eta(\ Machine\ 1)}{\eta\ (Machine\ 2)+(\ Machine\ 3)} \qquad \text{Equation 5.2}$$

This implies;

$$Prob\ (Job1\ for\ Machine\ 1) = \frac{0.00977}{0{,}0125+0.0026} \qquad \text{Equation 5.3}$$

$$= 0.64$$

In this way the probabilities are calculated and the job is assigned to the machine with the highest probability. In this formulation the values of α and β are taken as 1 and the pheromone up gradation function is not catered for initial convergence. The complete code is written in MATLAB and is attached in Appendix B. The same strategy was practiced on all 100 problem sets and the results are attached in Appendix C. As a sample the first five problems along with their detailed working are shown below. In addition a consolidated result of findings is given to summarize the effort.

Problem_ no 1

Jobs Assigned

Machine_1 = 1 4 5 8 10 11 12

Machine_2 = 2 3

Machine_3 = 6 7 9

Make-span = 70.5876 secs

Problem no = 2

Jobs Assigned

Machine_1 = 1 2

Machine_2 = 3 4 5 6 7 9 11

Machine_3 = 8 10 12

Make-span = 61.6470 secs

Problem no = 3

Jobs Assigned

Machine_1 = 2 5 7

Machine_2 = 1 4 6 8 9 10 11 12

Machine_3 = 3

Make-span = 77.0548 secs

<u>Problem no = 4</u>

Jobs Assigned

Machine_1 = 1 2 3 5

Machine_2 = 8 9 10 11

Machine_3 = 4 6 7 12

Make-span = 101.6560 secs

<u>Problem no =5</u>

Jobs Assigned

Machine_1 = 4 10 11

Machine_2 = 8 12

Machine_3 = 1 2 3 5 6 7 9

Make-span =107.3873 secs

The resultant make span obtained by this formulation is thus the value of $\boldsymbol{\psi_2}$.The summary of the results is presented in table 5.3 while addressing the following key factors:

- The % deviation between the result of ACO and other approaches such as SPT-A/LMC heuristics and GA technique with reference to the ideal solution from optimal (or exhaustive approach).
- The comparison of CPU time consumption to solve this set of 100 problems by each of the four approaches discussed in above paragraph.

Table 5. 3Summary of results of using ACO with other Techniques on a set of 100 problems

S No	Type of approach	Processor's speed	Time taken to solve all 100 problems	Average % Error from Exhaustive Search method/ Optimal Solution)
1	Exhaustive Search Method (all 4842288 possible options are to be explored)	P-IV, 2.0 GHz	680397 seconds (8 days)	-
2	Genetic Algorithm (GA)	P1V 1.7 GHz	7469 seconds (2 hrs)	0.20%
3	Hu's SPT-A/ LMC	P-IV, 2.0 GHz	300 seconds (5 mins)	5.84%
4	ACO	P-IV, 1.7 GHz	0.16 seconds	3.40%

5.4.2 Part-II: The ANN Domain of Formulation (Formulation ofψ_1)

One of the major advancements of this research is to develop an objective function which is capable of combining the two mechanisms i.e. the ACO and ANN technique. Thus the objective function should be able to cater for the convergence efficiency and should also be able to give ample reflection on the intelligent aspect of ANNs and future scenario building (as discussed in *chapter.no.4*). A run time or continuous solution building through intelligently analyzing and learning from the environment is taken as a new paradigm of optimized solution building. In this respect the concept of Artificial Neural Networks (ANNs) has emerged. This concept is based upon the ability of natural neurons to learn from repeated exposure to stimuli and then use the results of these past experiences to come up with a logical pattern through which they can predict future scenarios. The ability of ANNs to recognize the patterns, link and build relationships amongst likely unrelated entities through supervised or unsupervised learnings have inspired this formulation to develop a neural interface in the Pheromone Up gradation domain of ACO and then to use this interface to evolve a novel model which can not only be used in solving the scheduling problems but also it can be utilized to predict future scenarios/outputs. In this research a multilayered feed forward ANN is used to make it an integral part of basic ACO technique for evolution of ACO/Neural inspired model. This model has been able to combine the fast convergence characteristics of ACO with the pattern learning capability of ANNs.

The neural augmentation of ACO through the pheromone up-gradation component has been formulated on the ANNs having type of acyclic structures comprising of single or multiple hidden layers, whose calculation nodes are accordingly called neurons or units. The detailed analysis of these systems has already been covered in Chapter 4. Job of hidden neurons is interfacing between the extrinsic input and the structure output in a useful manner. The main function of the hidden layers is to take out repetitive values from the dense input given. The input vectors are fed forward to the hidden layers in between, till the ending layer which is called the output layer, gives genuine structure replication.

Thus ψ_1 is termed as "*programmer avec des imperfection*" or schedule with imperfections which has been formulated through the use of ANNs. Including this means that we the "high priority ground realities of the situation" are included as "C_{ij}" in terms of assigned variables. As in this research the assigned problem is formulated as follows:

$$\psi_1 = \sum_{\forall C_{ij}} (C_{ij}).(C_{IJ}^b) \qquad \text{Equation 5.4}$$

Here C_{IJ}^b is defined as

$$C_{IJ}^b = \begin{cases} 1, \text{ if } C_{ij} \text{ is integrated into the schedule} \\ 0 \text{ otherwise} \end{cases} \qquad \text{Equation 5.5}$$

And

$C_{ij} = w_1T_1. + w_2T_2 + w_3T_3 + \ldots\ldots.. w_iT_i$ Equation 5.6

Where w_i represents the weights assigned to machine processing times T_i .The function C_{ij} calculates the total ground realities or deviations as per the weighted priorities. Thus it is possible to control the contribution of different scheduling variables. For this research the weights are chosen to be equal to 1. This in itself can present a vast area of future research as the variation in the weighted priorities can significantly alter the results and can present intriguing possibilities.The emphasis on weighted inputs is because the scheduling problem can request many high priority variations and thus there has to be a "fairness" mechanism in order to justify the inclusion. This fairness mechanism has to consider the overall "build-up" of the problem itself. This buildup of the problem is not possible if the idealistic solution is not catered for. In order to bring intelligence to ψ_1 and to formulate a mechanism where it is capable of absorbing the ground realities and be realistic at the same time the research focuses on developing a ***neural aspect*** of this variable (as described in Chapter 4ANNs are capable of giving realistic and intelligent solutions while keeping in mind various different variables as inputs).

As discussed in Part-I of the formulation the set of 100 problems have been used for validation and building up of the complete NaACO model. In this respect there are three variables to start with i.e. A, B and E representing three processing times for machines 1,2 and 3. In order to develop a neural environment a mechanism is developed in which the ground realities are accounted for and at the same time the mechanism is intelligent enough to present a realistic solution. A model is constructed with a simple neural mechanism and the firing state of equation 5.6 is denoted by a single binary variable S.

If it fires then S=1 and if it doesn't fire then S=0. In this case the input is written as:

Input = $w_AS_A + w_BS_B + w_ES_E$ Equation 5.7

Where;

- S_A is the binary variable designated for decision variable A
- S_B is the binary variable designated for decision variable B and;
- $S_C$is the binary variable designated for decision variable C.

Here it is emphasized that these decision variables can assume different values as per the machines/ workstations/service centers etc. In addition to assigning neural sense to A, B and E the jobs which are fed into the mechanism are also considered as inputs. As such the complete input function for this problem can be written in form of equation 5.6 as;

Input = $w_A S_A + w_B S_B + w_E S_E + w_J S_J$ Equation 5.8

The weights assigned to these binary variables can vary from 0 to 1. These weights denomination is referent to the importance given to any one variable in relation with the other variables. The equal assignment of weights enable a fair initial or starting sequence in which all the variables are weighed equally. This equal denomination of weights construct the basics for this research to develop a convergent and a realistic mathematical model to handle scheduling problem. The input queries for the neuron triggering is generalized as;

$S_i \rightarrow 1$: $input \rightarrow input + w_i$ Equation 5.9

As such the neural environment shall work on the annotations of *Excitation* or *Inhibition*. If *input*↑ the environment shifts towards Excitation and if *input*↓ then we shift towards *Inhibition*. As far as the output on which the neural network is to be trained is concerned, a value is assigned to it which is close to the best results. This situation not only makes the solution effective but efficient as well.

In order to formulate the neural environment for ACO the pheromone up-gradation variable Tau has been focused. The reason for this selection is explained in Chapter 4. The pheromone upgradation mechanism in ACO has been used to reconsolidate the efforts of the ants. In this research the reconsolidation property of this function is further extended towards the intended acquisition of the forecasting property. The physical pheromone is the chemical which is extracted by the ants as they wander in the search of food sources. The pheromone in physical ants is different than from artificial ants which make use of this mechanism to validate their solutions and search for global optimas. To reiterate this point, the following table shows the selection of inputs for Tau in various ACO variants.

Table 5. 4Values for pheromone updation in various ACO variants.

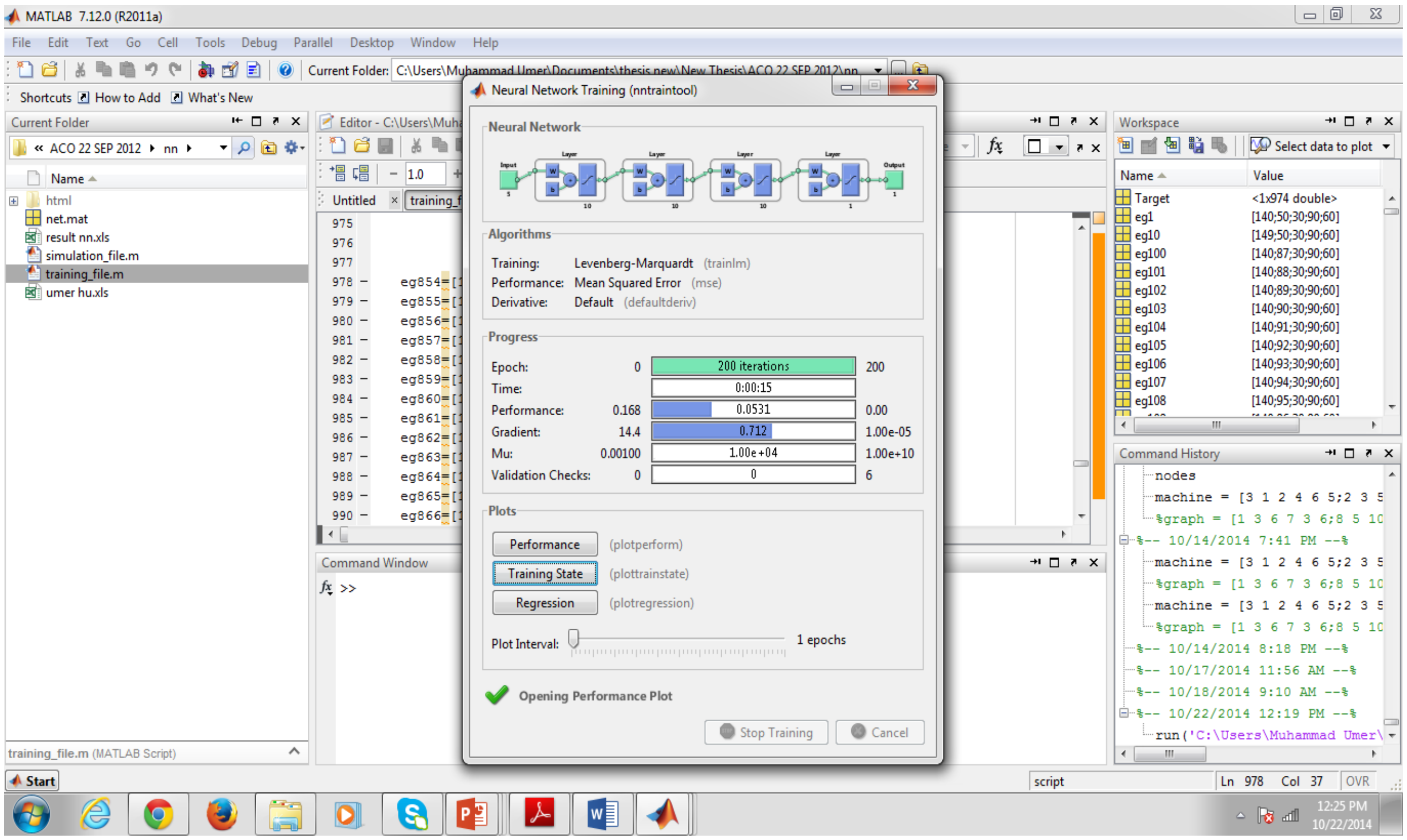
S No	ACO Variant	α	β	ρ	**M**	τ_o
1	AS	1	$2\ to\ 5$	0.5	N	m/C^{nn}
2	EAS	1	$2\ to\ 5$	0.5	N	$(e+m)/\rho C^{nn}$
3	AS_{rank}	1	$2\ to\ 5$	0.1	N	$0.5r(r-1)/\rho C^{nn}$
4	MMAS	1	$2\ to\ 5$	0.02	N	$1/\rho C^{nn}$
5	ACS	-	$2\ to\ 5$	0.1	10	$1/nC^{nn}$

Where m denotes the number of jobs and n denotes the number of machines to be visited. The given values of the parameters are for the reference as these values give good solutions. By looking at the composition of Tau, it is deduced that it is composed of inputs pertaining to the entire problem for which one wants to solve through ACO. Thus if this variable and the associated inputs are used to train an ANN with the outputs, the ACO can be augmented to forecast and predict for the future scenarios. For the scheduling problem in discussion Tau is calculated as per table 5.5

Table 5. 5Values of Tau1, Tau2, Tau3, calculated on Machine 1, 2, and 3.

	Machine 1					Machine 2					Machine 3				
Job	A_i	B_i	E_i	a+b/e	tau1	A_i	B_i	E_i	a+b/e	tau2	A_i	B_i	E_i	a+b/e	tau3
1	3	596	6	102.3333333	0.029315961	6	587	8	79.375	0.037795276	1	767	2	384.5	0.007802
2	2	70	9	9.777777778	0.306818182	7	221	1	228	0.013157895	1	433	9	49.11111	0.061086
3	1	507	6	85.5	0.035087719	1	285	5	58	0.051724138	0	395	8	49.375	0.060759
4	5	570	4	147.5	0.020338983	8	346	7	57.42857143	0.052238806	7	778	7	118.1429	0.025393
5	2	12	7	3.714285714	0.807692308	2	755	8	96.375	0.031128405	5	174	4	48.5	0.061856
6	5	344	1	349	0.008595989	9	97	3	41.33333333	0.072580645	5	303	1	308	0.00974
7	3	321	7	48.85714286	0.061403509	0	516	9	57.33333333	0.052325581	2	316	2	160	0.01875
8	8	220	7	39.42857143	0.076086957	0	278	8	34.75	0.086330935	4	225	2	116.5	0.025751
9	4	788	5	161.6	0.018564356	7	83	7	18.85714286	0.159090909	7	402	5	87.4	0.034325
10	1	642	2	322	0.00931677	3	148	9	19.44444444	0.154285714	7	110	4	34.5	0.086957
11	6	556	3	191.3333333	0.015679443	4	62	8	11.75	0.255319149	3	413	1	416	0.007212
12	7	334	3	118.3333333	0.025352113	1	346	4	87.5	0.034285714	9	772	3	266.3333	0.011264

In the above mentioned table (5.5) tau1 is the value of Tau calculated for machine 1 while keeping in view all of the jobs, and tau2 and tau3 are for machines 2 and 3 respectively.The input set comprising of machine times of machine 1, 2 and 3; tau values of machine 1, 2 and 3 is then neutrally trained against the output set of make spans achieved in part-I. The network is trained through the employment of a 3-layered network for this model. The training is done as per the following scheme:



Neural Network Training (nntraintool)

Algorithms

Training:	Levenberg-Marquardt	(trainlm)
Performance:	Mean Squared Error	(mse)
Derivative:	Default	(defaultderiv)

Progress

Epoch:	0	200 iterations	200
Time:		0:00:15	
Performance:	0.168	0.0531	0.00
Gradient:	14.4	0.712	1.00e-05
Mu:	0.00100	1.00e+04	1.00e+10
Validation Checks:	0	0	6

Plot Interval: 1 epochs

Opening Performance Plot

Workspace

Name	Value
Target	<1x974 double>
eg1	[140;50;30;90;60]
eg10	[149;50;30;90;60]
eg100	[140;87;30;90;60]
eg101	[140;88;30;90;60]
eg102	[140;89;30;90;60]
eg103	[140;90;30;90;60]
eg104	[140;91;30;90;60]
eg105	[140;92;30;90;60]
eg106	[140;93;30;90;60]
eg107	[140;94;30;90;60]
eg108	[140;95;30;90;60]

Command History

```
nodes
machine = [3 1 2 4 6 5;2 3 5
%graph = [1 3 6 7 3 6;8 5 10
%-- 10/14/2014 7:41 PM --%
machine = [3 1 2 4 6 5;2 3 5
%graph = [1 3 6 7 3 6;8 5 10
machine = [3 1 2 4 6 5;2 3 5
%graph = [1 3 6 7 3 6;8 5 10
%-- 10/14/2014 8:18 PM --%
%-- 10/17/2014 11:56 AM --%
%-- 10/18/2014 9:10 AM --%
%-- 10/22/2014 12:19 PM --%
run('C:\Users\Muhammad Umer\
```

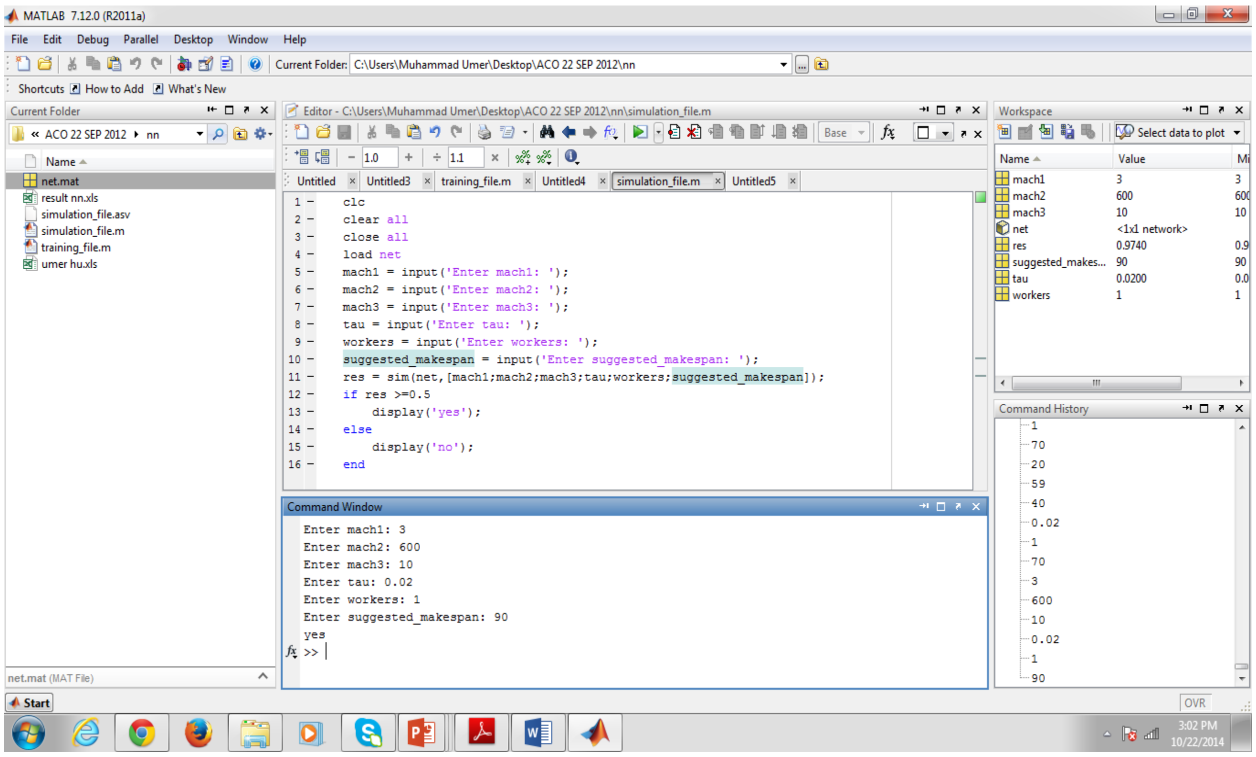
Figure 5. 3Training of data sets for NaACO

After training the network the simulation scheme is generated as follows:

```
par1 = input ('Enter Time Machine 1: ');
par2 = input ('Enter Time Machine 2: ');
par3 = input ('Enter Time Machine 3: ');
par4 = input ('Enter Requested Make-span: ');
Res = sim ([par1;par2;par3;par4]);
If res >= 0.8
    Display ('Yes');
Else
    Display ('No');
End
```

This simulation scheme (figure 5.5) prompts the user regarding the inputs pertaining to the current situation and gives a response in terms of feasibility of the combination or otherwise. If feasible the answer is “Yes” and if infeasible the answer is “No”. If the answer is “Yes”, the “make span” which has been accepted in the combination is the value of ψ_1. The objective is to “minimize” the difference between ψ_1 and ψ_2. Thus the objective function for this formulation is given as:

Minimize $\psi(\sigma) = \psi_1 - \psi_2$ Equation 5.10



```
clc
clear all
close all
load net
mach1 = input('Enter mach1: ');
mach2 = input('Enter mach2: ');
mach3 = input('Enter mach3: ');
tau = input('Enter tau: ');
workers = input('Enter workers: ');
suggested_makespan = input('Enter suggested_makespan: ');
res = sim(net,[mach1;mach2;mach3;tau;workers;suggested_makespan]);
if res >=0.5
    display('yes');
else
    display('no');
end
```

```
Enter mach1: 3
Enter mach2: 600
Enter mach3: 10
Enter tau: 0.02
Enter workers: 1
Enter suggested_makespan: 90
yes
>>
```

Name	Value
mach1	3
mach2	600
mach3	10
net	<1x1 network>
res	0.9740
suggested_makes...	90
tau	0.0200
workers	1

Figure 5. 4Obtaining value forψ_1 in NaACO

Thus for the five instances described in the formulation part-I earlier the values of ψ_1 are 70.58, 61.64, 77.054, 101.65 and 107.38 respectively. The suggested make span in the figure is the value of ψ_2. Hence the value of the objective function is: 90 – 70.58 = 19.42. The value of the objective function is measured against the real time gap and leverage available at the scheduling floor and the priority set by the operators. The above model is thus referred as a continuous time model, as it considers a stochastic behavior of a single job. The complete formulation is termed as **NaACO (Neural Augmented ACO).**The suggested approach is capable of handling very large data sets and is able to return the accurate results within a very short span of time. This makes NaACO unique as it has given forecasting and forward thinking capabilities to ACO while keeping the fast convergence facet of conventional ACO in hand.

5.4.3 Complete Formulation of NaACO

A complete mathematical formulation model for NaACO is summarized as follows:

To schedule n jobs to m parallel machines;

For each job represented by j with the total number of jobs as n, the constraints of the problem are;

$\sum_{i=1}^{n} j_i \in m_1 \; or \; m_2 \; or \; m_3 \; \ldots\ldots m_n$ at any time t (1)

Constraint (1) refers to the notion that each job has to be designated to any of the machines at any time. The complete designation is mandatory.

$\forall j_{i\,(i=1,2,,.n)} \notin (m_1, m_2), (m_1, m_3), (\, m_2, m_3) \ldots (\, m_i, m_j)$ (2)

Constraint (2) addresses that for all jobs considered there is no job splitting allowed i.e. the jobs cannot be split in between any operation once the work has already started on a particular job.

$m_{t1} + \; m_{t2} + \; m_{t3} + \cdots \; m_{tn} \leq \; m_{Tt}$ (3)

Constraint (3) iterates that the sum of the total processing times of each machines for each job cannot exceed the total processing time of all machines combined.

$m_{t1}, m_{t2}, m_{t3}, \ldots m_{tn} > 0$ (4)

Constraint (4) reflects on the constraint that the machine processing times adhere to the non-negativity constraint.

$m_{t(i)setup} \; \approx 0$ (5)

The setup times as depicted in constraint (5) for the machines which are to be scheduled are negligible. These times are catered for in the total processing times for each machine.

$$m_{t(i)transp} \approx 0 \qquad (6)$$

The transportation time between the machines is negligible as is explained in constraint (6).

$$\forall J, (J = 1 \ldots . n); \sum_{i=1}^{n} P_i = P.\mathrm{T} \qquad (7)$$

Form condition (7) the total process time (P.T) is the sum of all individual process times for all jobs which are scheduled on all the machines.

To further reiterate the constraints in problem formulation the first constraint is regarding the job allotment to machines. It explains that each job has to belong to any one machine at any given time moreover once the jobs are assigned to a machine they cannot be removed before their completion.The jobs are constrained in this fashion that they cannot belong to two or more machines at one time, hence the jobs cannot be split and moreover each job has to go to one machine. If this situation is altered this presents yet another paradigm for future research in which the job splitting situation greatly increase the applicability of NaACO for FJSSP. The total machine time cannot exceed the total time taken by the all machines involved, which means that each machine has to adhere to the total process time limitation. This also reflects the tolerance available with the scheduler to cater for the minimum and maximum time. Moreover the machine setup times and the machine transportation times are also two variables which can be earmarked for further application of NaACO in terms of the determination of allowances available in these two domains while predicting for the best realistic solution.

After description of the constraints now the problem is formulated as;

Minimize $\quad \psi\ (\sigma)_{=} \psi_1 - \psi_2$

$Where$:

$$\sigma = \sum_{i=1}^{n} P_i$$

And

$$\psi_1 = \sum_{\forall C_{ij}} (C_{ij}).(\,C_{IJ}^b)$$

Here ψ1 is the on-ground based solution and

$\psi_2 = ACO\ inspired\ Ideal\ Solution$

For formulation of ψ1the ANN formulation of the problem as follows:

C_{IJ}^b = 1, if C_{ij} is integrated into the schedule

0 otherwise

The weights for times are assigned as;

$C_{ij} = w_1T_1. + w_2T_2 + w_3T_3 + \ldots\ldots.. w_iT_i$

And triggering of:

$S_i \to 1$: $input \to input + w_i$

The constraints of the problem are summarized as;

$\sum_{i=1}^{n} j_i \in m_1\ or\ m_2\ or\ m_3$at any time t (1)

$\forall j_{i\ (i=1,2,,,n)} \notin (m_1, m_2), (m_1, m_3), (\,m_2, m_3) \ldots.\ (\,m_i, m_j)$ (2)

$m_{t1} + m_{t2} + m_{t3} + \cdots m_{tn} \leq m_{Tt}$ (3)

$m_{t1}, m_{t2}, m_{t3} > 0$ (4)

$m_{t(i)setup} \approx 0$ (5)

$m_{t(i)transp} \approx 0$ (6)

$\forall J, (J = 1 \ldots.n); \sum_{i=1}^{n} P_i = P.\mathrm{T}$ (7)

5.5 Chapter Summary

This chapter can be summarized as follows:

- The concept of NaACO is mathematically modelled in this chapter. The objective function is generated keeping in view the ACO and the ANN domains.
- The NaACO technique is formulated according to the steps mentioned in the chapter. Moreover, each step is also validated at the time of formulation by implementing it on a set of 100 problems found in the literature. These sets provide necessary formulation foundations for NaACO.
- The results of the formulation are also matched with the results so as to make sure that the formulation is up to mark and is validated.
- This research is focused on the application of ACO and then how to use a convergence technique to formulate a Neural Augmented ACO. As regards to this the initial convergence should give the ideal solution (hence $\boldsymbol{\psi_2}$) and then based on the corrective factor of ACO i.e. tau which takes into account the inputs from the problem sets, formulation of $\boldsymbol{\psi_1}$is accomplished. In this research the neural augmentation of ACO gives an approach labelled as NaACO (Neural Augmented ACO) to include the intelligent aspect of ACO for solving the problems in scheduling/ manufacturing and services environments.

CHAPTER 6: VALIDATION OF NaACO

6.1 Preamble

In this chapter the NaACO technique has been validated through its application on key benchmark problems. The overall validation process is explained in the validation roadmap as depicted in figure 6.1. As shown in this figure (6.1) the validation process comprises of firstly feeding the problem to the NaACO technique thorough ACO domain and attaining the ideal results. Then the results are used to trigger the ANN domain in order to give the realistic value. As already explained in Chapter 5 the ideal solution is termed as ψ_2 and the realistic solution through ANN is termed as ψ_1. The validation approach of ANN domain is unique as it is validated for a solution space in which the solution is already known and the ANN component is prompted to give a futuristic forecast so as to match the results with the already known values.

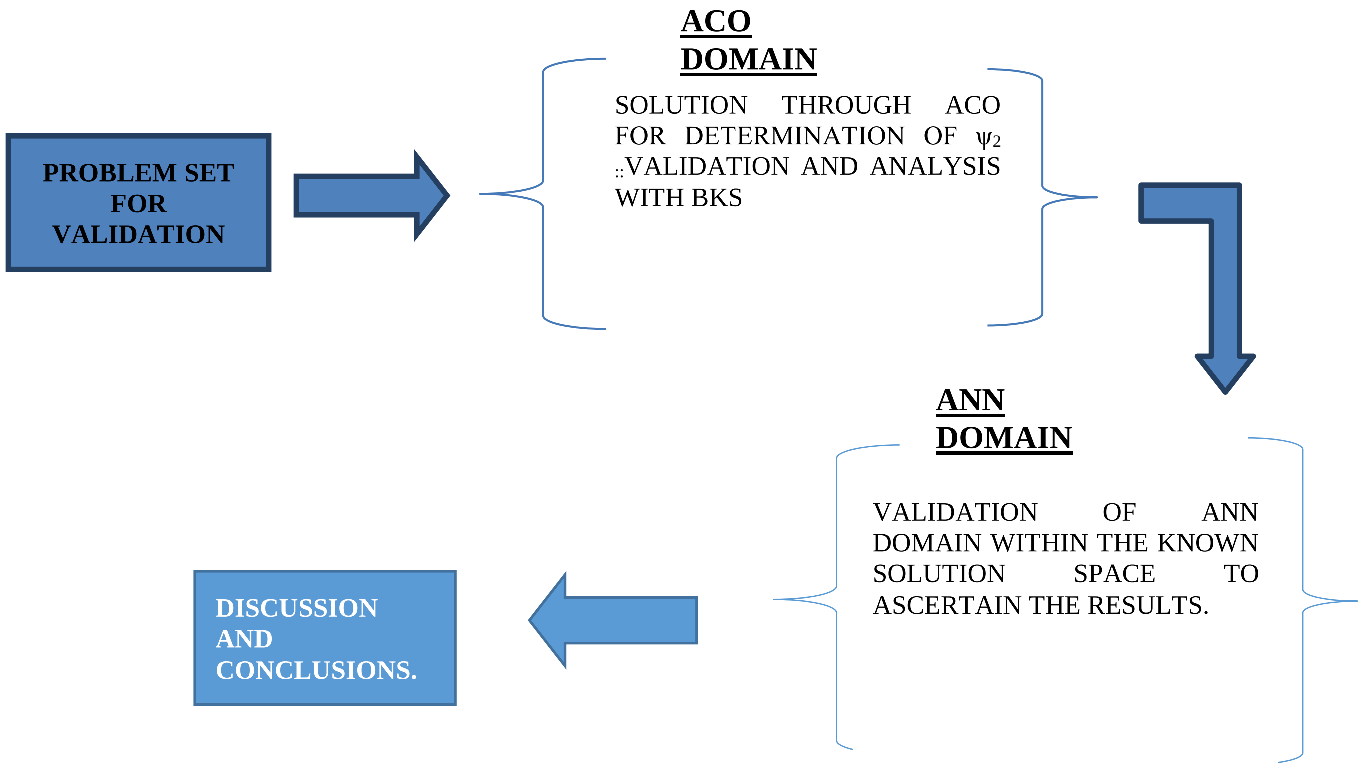

PROBLEM SET FOR VALIDATION
ACO DOMAIN
SOLUTION THROUGH ACO FOR DETERMINATION OF ψ_2 ::VALIDATION AND ANALYSIS WITH BKS
ANN DOMAIN
VALIDATION OF ANN DOMAIN WITHIN THE KNOWN SOLUTION SPACE TO ASCERTAIN THE RESULTS.
DISCUSSION AND CONCLUSIONS.

Figure 6. 1NaACO Validation Roadmap

The NaACO approach is validated through application of the methodology on a set of bench mark problems. The problems selected for this purpose are following:

- FT06
- FT10
- FT20

These problems are verified benchmark problems which have known make-spans and answers. Out of these problem sets, FT10 and FT20 are considered to be true NP-Hard problem sets. The validation approach consists of the formulation of ACO approach to find the ideal minimum makespan. Once done, the NaACO approach is formulated to verify the answers from a set of known data sets in order to validate the complete approach. The validation approach on each of the problem sets is discussed in the following sections of this chapter with both the domains of NaACO being catered for separately as per figure 6.1.

6.2 **FT06:**

A typical FT06 problem comprises of scheduling six jobs on six machines (the complete formulation code of NaACO in MATLAB for FT06 is given in Appendix B). The general assumptions of the problem are:

- No preemption. This assumption means that the completion of a job on a particular machine shall qualify that machine for the next job. There is no breakdown. This means that from the start till end of all the operations machines do not encounter any malfunctions.
- The setup time is included in the overall machine time.
- The inclusion of any new job beyond the specified jobs is not allowed. It means that the set of specified timings and the number of jobs is fixed and shall remain unchanged throughout the problem.

- The priority principal right at the onset is not catered; i.e. the jobs can go to any machine at the start of the scheduling process.
- The time to inspect the jobs is included in the overall times for the jobs as specified in the problem set.
- The transportation time is negligible.

Table 6. 1FT06 Problem Set

Jobs	machine,time	machine,time	machine,time	machine,time	machine,time	machine,time
1	**3,1**	**1,3**	**2,6**	**4,7**	**6,3**	**5,6**
2	**2,8**	**3,5**	**5,10**	**6,10**	**1,10**	**4,4**
3	**3,5**	**4,4**	**6,8**	**1,9**	**2,1**	**5,7**
4	**2,5**	**1,5**	**3,5**	**4,3**	**5,8**	**6,9**
5	**3,9**	**2,3**	**5,5**	**6,4**	**1,3**	**4,1**
6	**2,3**	**4,3**	**6,9**	**1,10**	**5,4**	**3,1**

6.2.1 ACO Domain

The above mentioned problem set (table 6.1) is a benchmark FT06 problem set. As per the methodology firstly ACO is applied to attain the value of $\psi_{2.}$ The following relationship is used to setup a triggering heuristic function:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l\in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \text{ ; If j} \in N_i^k \qquad \text{Equation 6.1}$$

In the above expression the value of τ_{ij} is the reciprocal of the heuristic function. In this case the heuristic function is the shortest processing time for each job. The values of alpha and beta are kept as 1. Moreover the pheromone upgrade for the first iteration is not utilized. It is later on utilized to generate sequences of schedules. The sequences as generated by this method are scattered to avoid any duplication of jobs assigned to a particular machine. The sequences in this particular case for job splitting are demonstrated in figure 6.2

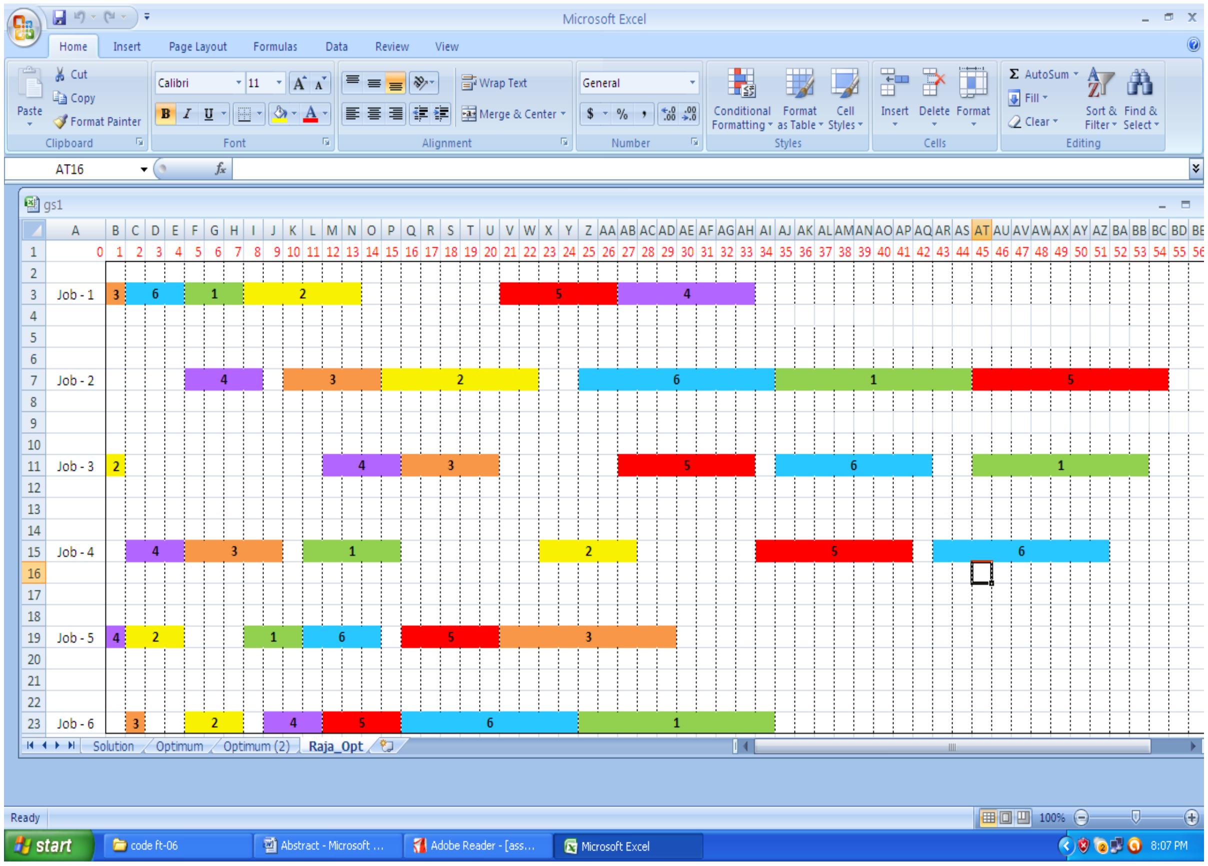
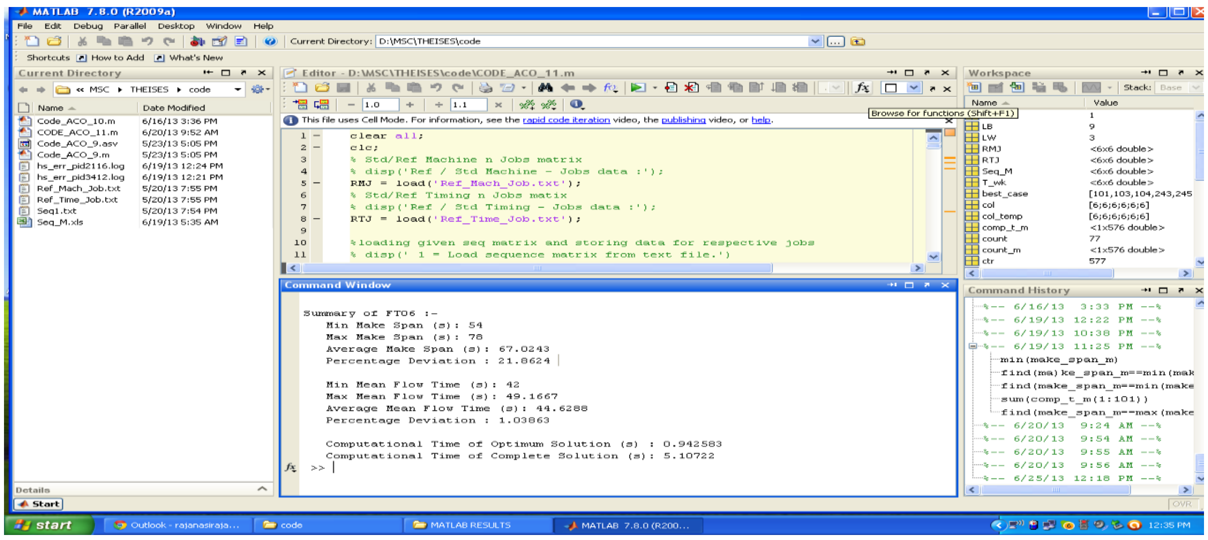


Figure 6. 2Job splitting sequences of FT06

The makespan for the above mentioned problem is then coded with the following results:


MATLAB 7.8.0 (R2009a)
File Edit Debug Parallel Desktop Window Help
Current Directory: D:\MSC\THEISES\code
Shortcuts How to Add What's New
Current Directory
Editor - D:\MSC\THEISES\code\CODE_ACO_11.m
Workspace
Command Window
Command History
start

Name	Date Modified
Code_ACO_10.m	6/16/13 3:36 PM
CODE_ACO_11.m	6/20/13 9:52 AM
Code_ACO_9.asv	5/23/13 5:05 PM
Code_ACO_9.m	5/23/13 5:05 PM
hs_err_pid2116.log	6/19/13 12:24 PM
hs_err_pid3412.log	6/19/13 12:21 PM
Ref_Mach_Job.txt	5/20/13 7:55 PM
Ref_Time_Job.txt	5/20/13 7:55 PM
Seq1.txt	5/20/13 7:54 PM
Seq_M.xls	6/19/13 5:35 AM

This file uses Cell Mode. For information, see the rapid code iteration video, the publishing video, or help.

```
clear all;
clc;
% Std/Ref Machine n Jobs matrix
% disp('Ref / Std Machine - Jobs data :');
RMJ = load('Ref_Mach_Job.txt');
% Std/Ref Timing n Jobs matix
% disp('Ref / Std Timing - Jobs data :');
RTJ = load('Ref_Time_Job.txt');

%loading given seq matrix and storing data for respective jobs
% disp(' 1 = Load sequence matrix from text file.')
```

```
Summary of FT06 :-
   Min Make Span (s): 54
   Max Make Span (s): 78
   Average Make Span (s): 67.0243
   Percentage Deviation : 21.8624

   Min Mean Flow Time (s): 42
   Max Mean Flow Time (s): 49.1667
   Average Mean Flow Time (s): 44.6288
   Percentage Deviation : 1.03863

   Computational Time of Optimum Solution (s) : 0.942583
   Computational Time of Complete Solution (s): 5.10722
>>
```

Name	Value
	1
LB	9
LW	3
RMJ	<6x6 double>
RTJ	<6x6 double>
Seq_M	<6x6 double>
T_wk	<6x6 double>
best_case	[101,103,104,243,245
col	[6;6;6;6;6;6]
col_temp	[6;6;6;6;6;6]
comp_t_m	<1x576 double>
count	77
count_m	<1x576 double>
ctr	577

```
%-- 6/16/13  3:33 PM --%
%-- 6/19/13 12:22 PM --%
%-- 6/19/13 10:38 PM --%
%-- 6/19/13 11:25 PM --%
   min(make_span_m)
   find(ma)ke_span_m==min(mak
   find(make_span_m==min(make
   sum(comp_t_m(1:101))
   find(make_span_m==max(make
%-- 6/20/13  9:24 AM --%
%-- 6/20/13  9:54 AM --%
%-- 6/20/13  9:55 AM --%
%-- 6/20/13  9:56 AM --%
%-- 6/25/13 12:18 PM --%
```

Figure 6. 3Results achieved by coding in MATLAB.

Table 6. 2Results of ACO domain

Parameters	**BKS**	**Proposed Algorithm**
Makespan (sec)	55	**55**
Mean Flow Time (sec)	44.17	**42.33**
Jobs Delay Time (sec)	-	57
Machines Idle Time (sec)	-	50
Total Dormancy Time (sec)	-	77

The minimum makespan is found to be 55. The percentage deviation is 1.03% in average flow time. These results are also looked at in response to the computational time which ACO takes to reach a solution which is far less than the counterpart techniques being used. The result is summarized as follows in table 6.3:

Table 6. 3Results of ACO domain

Optimization Meta-heuristic	**Makespan**	**Computational Time (seconds)**
Proposed ACO	**55**	**0.48**
GA	55	1
PSO	55	7
TS	55	2
Conventional CSA	55	50
PSMCSA	55	17

The algorithm achieved the best values of minimum make span for seven times as shown in the following history of iterations:

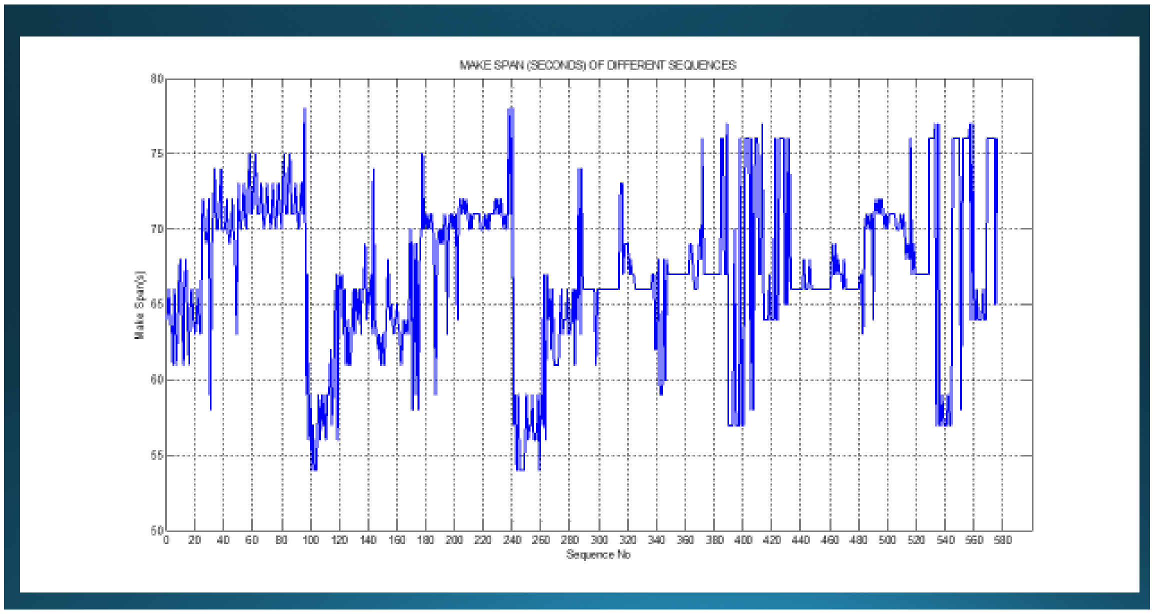
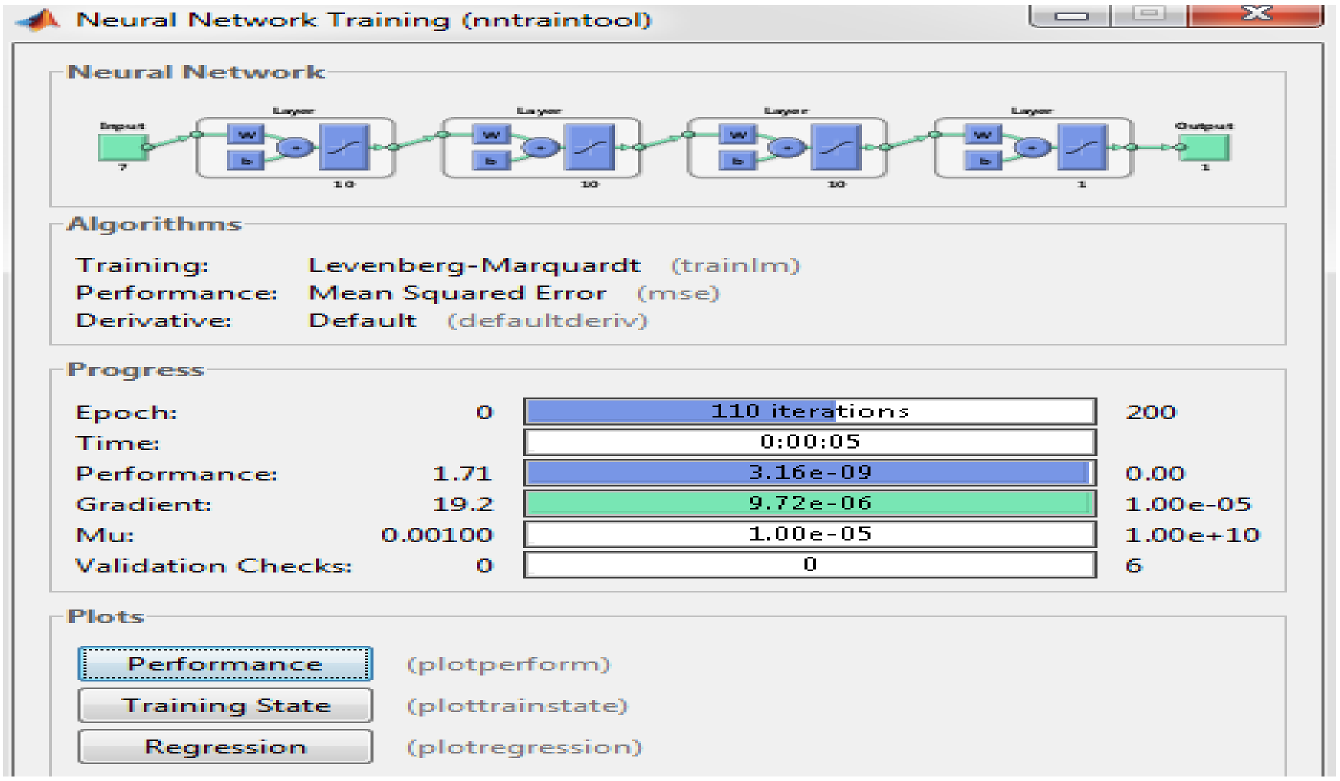


Figure 6. 4Iteration sequences of FT06

6.2.2 ANN Domain

For this problem the value of ψ_2 comes out to be 55 seconds. In order to validate the NaACO approach initially the data set is validated on the known values i.e. the results are stopped in between and the ANN segment is trained on the known values of inputs, tau and outputs. The sequences are then tested to ascertain that the complete NaACO formulation against the known results. In this manner the model is firstly validated within the known set of results. The following results in figure 6.5 depict this situation.



Algorithms

Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)

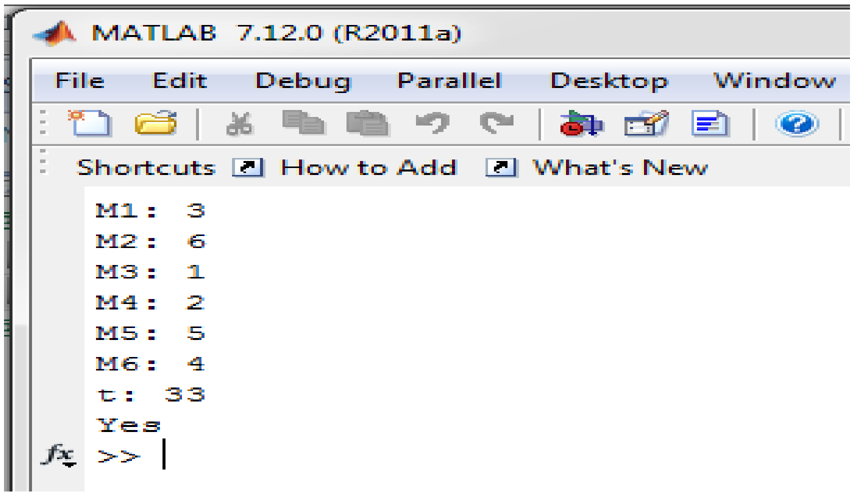
Progress

Epoch:	0	110 iterations	200
Time:		0:00:05	
Performance:	1.71	3.16e-09	0.00
Gradient:	19.2	9.72e-06	1.00e-05
Mu:	0.00100	1.00e-05	1.00e+10
Validation Checks:	0	0	6

(plotperform)
(plottrainstate)
(plotregression)

Figure 6. 5Training sequences of FT06 for ANNs

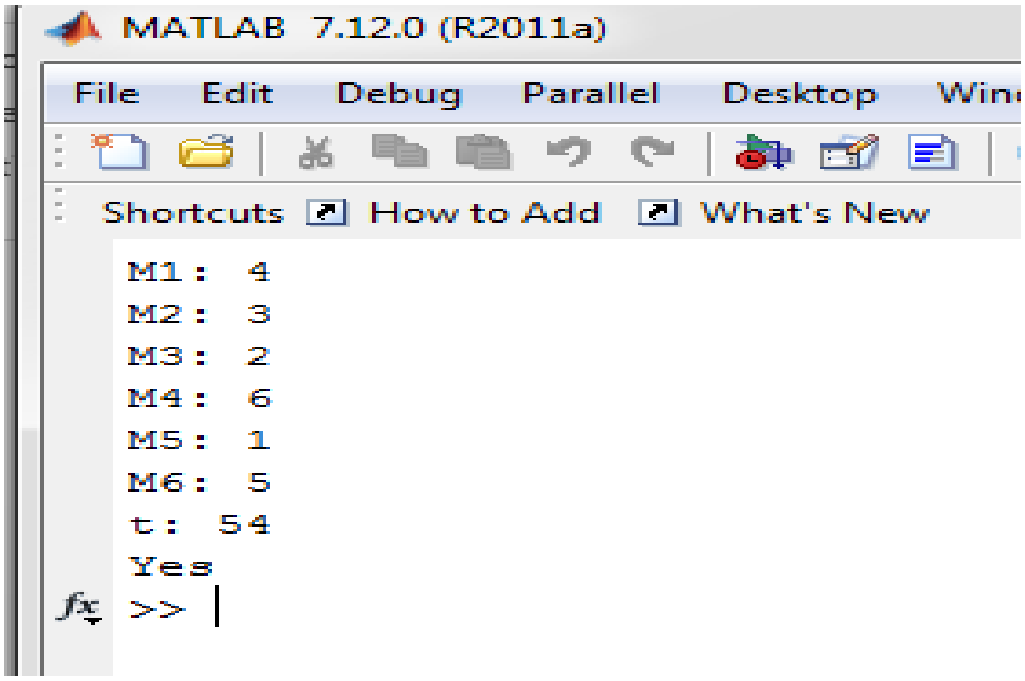
The above results train the NaACO model within the frame work of known parameters. The model is further validated once some arbitrary values are put into consideration to check the response. The following snapshots depict those situations:


MATLAB 7.12.0 (R2011a)
File Edit Debug Parallel Desktop Window
Shortcuts How to Add What's New

```
M1:  3
M2:  6
M3:  1
M4:  2
M5:  5
M6:  4
t:  33
Yes
>> |
```

Figure 6. 6ANN dialogue box for validation of NaACO

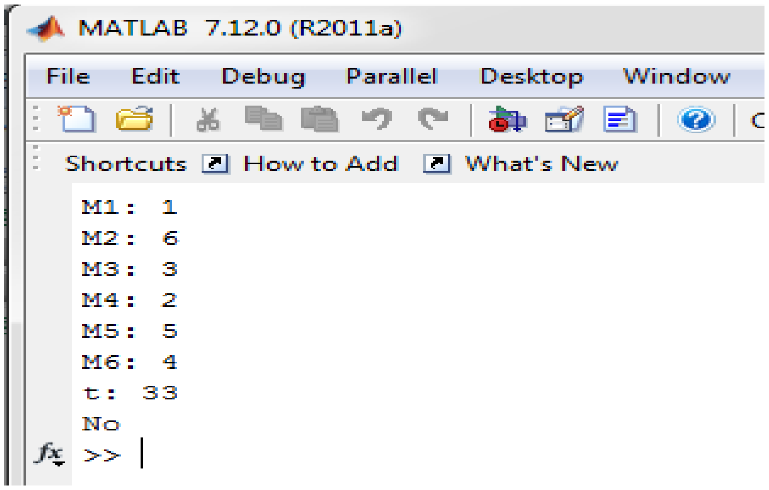
In figure 6.6 the NaACO returns a YES for a feasible combination within the known domain of answers which is used to validate the training in ANN domain.


MATLAB 7.12.0 (R2011a)
File Edit Debug Parallel Desktop Win
Shortcuts How to Add What's New

```
M1:  4
M2:  3
M3:  2
M4:  6
M5:  1
M6:  5
t:  54
Yes
>>
```

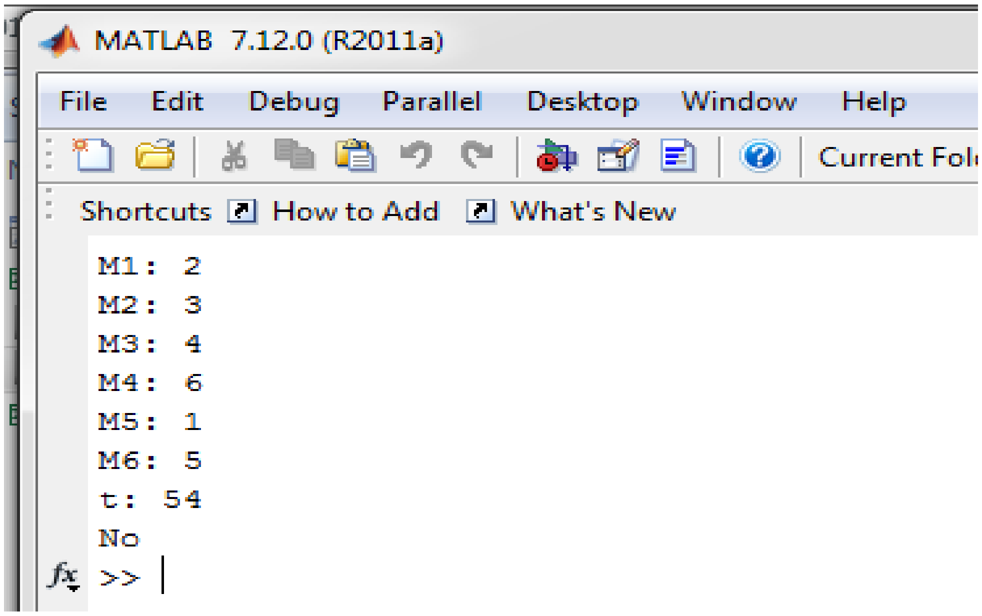
Figure 6. 7 ANN dialogue box for validation of NaACO. It returns a YES for validation on FT06.

The above results validate the NaACO model within the frame work of known parameters. The model is further tested once some arbitrary values are put into consideration to check the response. The situations in figures 6.8 and 6.9 depict those scenarios:


MATLAB 7.12.0 (R2011a)
File Edit Debug Parallel Desktop Window
Shortcuts How to Add What's New

```
M1:  1
M2:  6
M3:  3
M4:  2
M5:  5
M6:  4
t:  33
No
>>
```

Figure 6. 8NaACO returns a NO for a likely infeasible combination.


MATLAB 7.12.0 (R2011a)
File Edit Debug Parallel Desktop Window Help
Current Fol
Shortcuts How to Add What's New

```
M1:  2
M2:  3
M3:  4
M4:  6
M5:  1
M6:  5
t:  54
No
>> 
```

Figure 6. 9NaACO returns a NO for a likely infeasible combination

As depicted the sequence of assignment is changed from figure 6.7 to a new sequence in figure 6.8, NaACO has recognized the change and has returned a NO prompt which suggests that with the given processing times, sequences, pheromone values this sequence is not possible to execute. Once the model is validated within the known value sets the next step is to determine the value of ψ_1. This can be accomplished by putting in different processing times, in the original matrix of FT06 problem set and then training the new situation.

6.2.3 Discussion on Results

As a first validation instance for NaACO the FT06 problem has been addressed. The problem set has been subjected to the validation process as described in the beginning of this chapter. The results of this treatment show that NaACO has been able to achieve the initial convergence through ACO domain and has reached the BKS values. Moreover the ANN domain of NaACO also has validated the forecasting correctness of NaACO by producing the results within the known solution sets. The algorithm was able to achieve the BKS value in a very short span of time. Furthermore within the ANN domain it is pertinent to point out that the post optimal analysis of the machine sequences and of the makespan values during the training of NaACO so as to ascertain the limits for the trained ANNs to prompt the YES or NO responses can pose a vast area for future researchers of NaACO.

6.3 **FT10:**

The next problem set picked for model validation is FT10. This problem set comes under the domain of true NP-Hard problems. The problem is described as scheduling of 10 jobs on 10 machines with known processing times. The problem has a wider and more elaborated search space than the previously considered problem of FT06. This problem has been used to validate the NaACO approach as per the formulation of the validation process (the complete formulation code of NaACO in MATLAB for FT10 is given in Appendix B).

Table 6. 4Machine matrix for FT10

Mi/Ji	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
J1	1	2	3	4	5	6	7	8	9	10
J2	1	3	5	10	4	2	7	6	8	9
J3	2	1	4	3	9	6	8	7	10	5
J4	2	3	1	5	7	9	8	4	10	6
J5	3	1	2	6	4	5	9	8	10	7
J6	3	2	6	4	9	10	1	7	5	8
J7	2	1	4	3	7	6	10	9	8	5
J8	3	1	2	6	5	7	9	10	8	4
J9	1	2	4	6	3	10	7	8	5	9
J10	2	1	3	7	9	10	6	4	5	8

Table 6. 5Time matrix for FT10

Ji/Mi	MI	M2	M3	M4	M5	M6	M7	M8	M9	M10
J1	29	78	9	36	49	11	62	56	44	21
J2	43	28	90	69	75	46	46	72	30	11
J3	85	91	74	39	33	10	89	12	90	45
J4	71	81	95	98	99	43	9	85	52	22

J5	6	22	14	26	69	61	53	49	21	72
J6	47	2	84	95	6	52	65	25	48	72
J7	37	46	13	61	55	21	32	30	89	32
J8	86	46	31	79	32	74	88	36	19	48
J9	76	69	85	76	26	51	40	89	74	11
J10	13	85	61	52	90	47	7	45	64	76

For the said problem, the methodology is to assign 10 jobs on 10 machines through ACO to obtain ψ_1 and then to validate the NaACO through neural training of the value obtained through calculation of Tau. The general assumptions of this problem are the same as of FT06 except that now the jobs and machines have increased from 6 to 10. The validation is done within the preview of the calculated results to demonstrate the accuracy of ANN. The validation is subdivided into two parts i.e. the ACO domain of the problem and the ANN domain of the problem. The validation results are summarized at the end of the section.

6.3.1 ACO Domain:

The ACO formulation of FT10 is explained as follows:

- The reciprocal of the heuristic function is calculated based on the greedy heuristic of minimum processing time. This implies that for this problem η = 1/ min.processing time. The values are as per table 6.6:

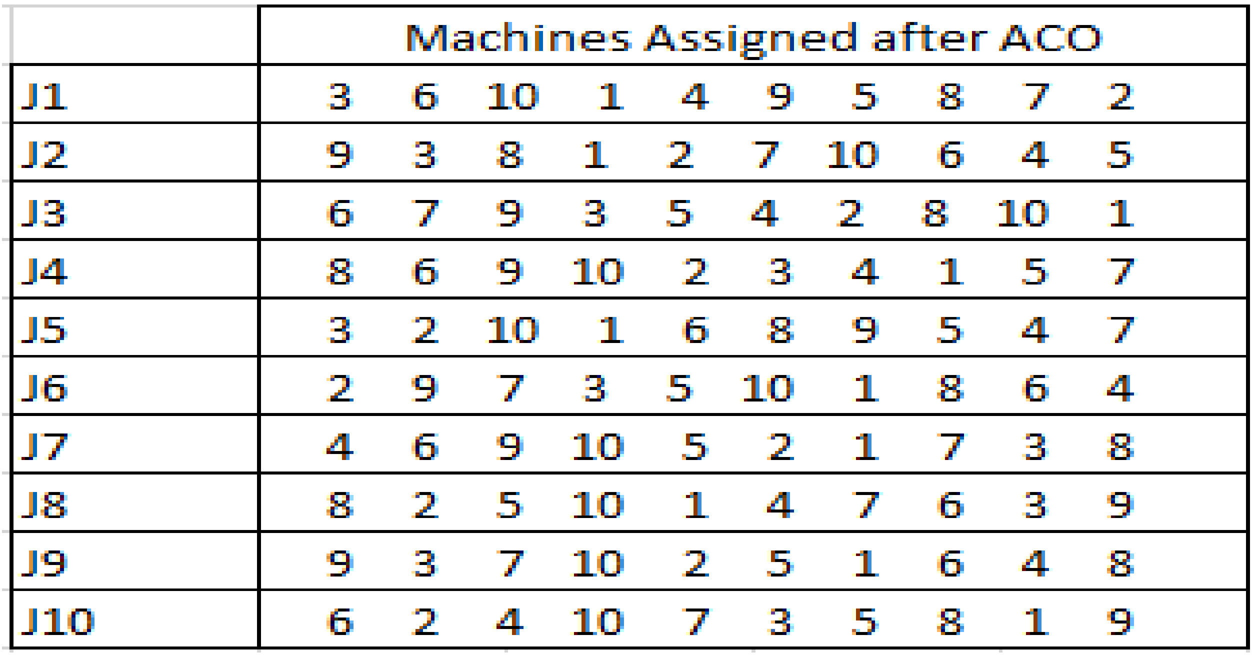
Table 6. 6Values of η for FT10

0.034483	0.012821	0.111111	0.027778	0.020408	0.090909	0.016129	0.017857	0.022727	0.047619
0.023256	0.035714	0.011111	0.014493	0.013333	0.021739	0.021739	0.013889	0.033333	0.090909
0.011765	0.010989	0.013514	0.025641	0.030303	0.1	0.011236	0.083333	0.011111	0.022222
0.014085	0.012346	0.010526	0.010204	0.010101	0.023256	0.111111	0.011765	0.019231	0.045455
0.166667	0.045455	0.071429	0.038462	0.014493	0.016393	0.018868	0.020408	0.047619	0.013889
0.021277	0.5	0.011905	0.010526	0.166667	0.019231	0.015385	0.04	0.020833	0.013889

0.027027	0.021739	0.076923	0.016393	0.018182	0.047619	0.03125	0.033333	0.011236	0.03125
0.011628	0.021739	0.032258	0.012658	0.03125	0.013514	0.011364	0.027778	0.052632	0.020833
0.013158	0.014493	0.011765	0.013158	0.038462	0.019608	0.025	0.011236	0.013514	0.090909
0.076923	0.011765	0.016393	0.019231	0.011111	0.021277	0.142857	0.022222	0.015625	0.013158

- Based on these values, the probabilities of each job to be assigned are calculated by incorporating the fundamental ACO equation with no pheromone upgrade to give a sequence of machines and a sequence of sorted times as given in table 6.7.

Table 6. 7Machine sorting for FT10



	Machines Assigned after ACO									
J1	3	6	10	1	4	9	5	8	7	2
J2	9	3	8	1	2	7	10	6	4	5
J3	6	7	9	3	5	4	2	8	10	1
J4	8	6	9	10	2	3	4	1	5	7
J5	3	2	10	1	6	8	9	5	4	7
J6	2	9	7	3	5	10	1	8	6	4
J7	4	6	9	10	5	2	1	7	3	8
J8	8	2	5	10	1	4	7	6	3	9
J9	9	3	7	10	2	5	1	6	4	8
J10	6	2	4	10	7	3	5	8	1	9

- The next step is the sequencing of machines on the basis of no overlapping and then spreading the machines to avoid multiprocessing. This is done by sorting the probabilities of ACO so as to assign the jobs to the machine having the least processing time. If there is a tie then the priority goes to the machine having the greater probability. The code is

• **Parameters**	• **BKS**	• **Proposed Algorithm**
• Make span (sec)	• 930	• **930**
• Avg Make span (sec)	• 1171 (SPT)	• **1370**
• % Deviation (Avg make span on SPT)		• 14.52 %

programmed in MATLAB and shows that ACO was able to achieve the minimum makespan of 930 and an average makespan of 1370 with an error of 14.52% (the average SPT value for FT10 is 1171). The discussion on this aspect is done in sub section (6.3.3). The summary of the results obtained for ACO domain is presented in table 6.8 below.

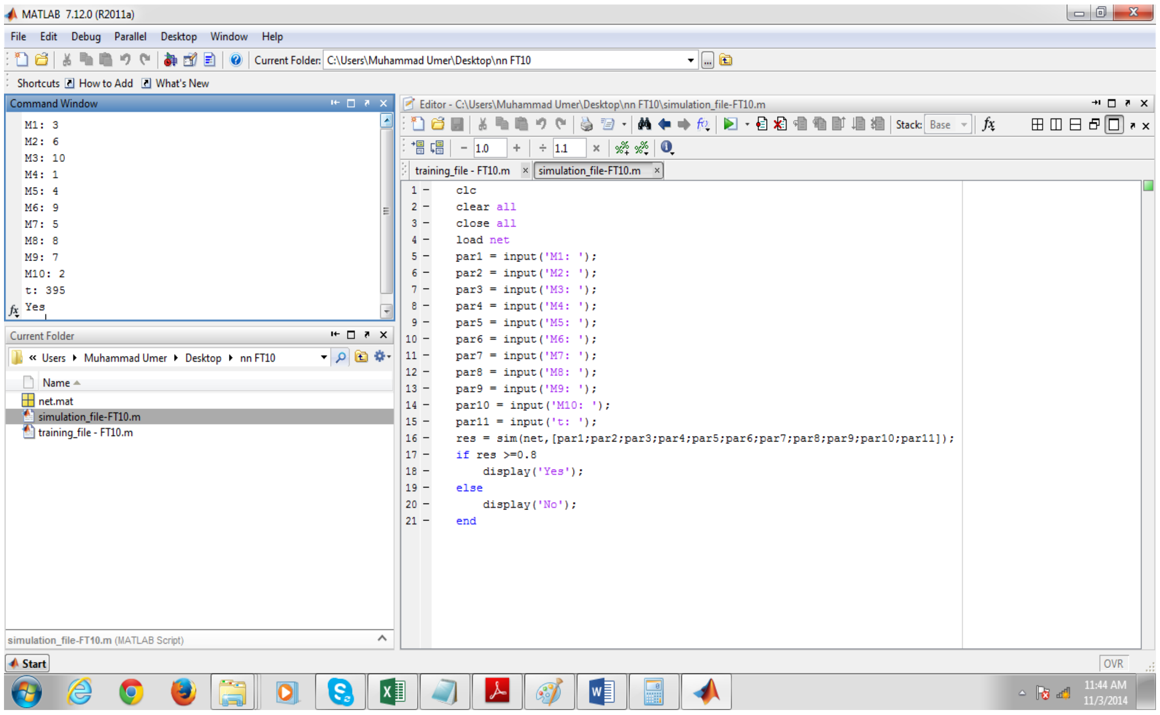
6.3.2 ANN Domain:

With this sequence the value for $\psi_1$is obtained. For the validation of NaACO the model is trained on inputs, tau values and outputs. The training is done to verify the results within the known domain of answers. Firstly the system is customized to give tau values and the average tau value as follows:

Table 6. 8tau values for ANN training of the problem set.

	Machines Assigned after ACO										Tau
J1	3	6	10	1	4	9	5	8	7	2	0.025316
J2	9	3	8	1	2	7	10	6	4	5	0.021552
J3	6	7	9	3	5	4	2	8	10	1	0.017606
J4	8	6	9	10	2	3	4	1	5	7	0.015267
J5	3	2	10	1	6	8	9	5	4	7	0.025445
J6	2	9	7	3	5	10	1	8	6	4	0.020161
J7	4	6	9	10	5	2	1	7	3	8	0.024038
J8	8	2	5	10	1	4	7	6	3	9	0.018553
J9	9	3	7	10	2	5	1	6	4	8	0.01675
J10	6	2	4	10	7	3	5	8	1	9	0.018519
									Avg=		0.020321

This value of tau along with the inputs of machine sequence, and make-span for each combination is utilized to constitute an ANN which is trained on the current system. This training is accomplished through supervised learning of inputs versus the outputs. At the termination of training the simulations are carried out to ascertain the validity of the model within the defined and known answers. The following figures depict a correct combination of machines and time and an incorrect combination of the machine sequence and times to show the validity of the model within the specified domains. In figure 6.10 the machine sequence is accepted with the time given and the response of the ANN is a

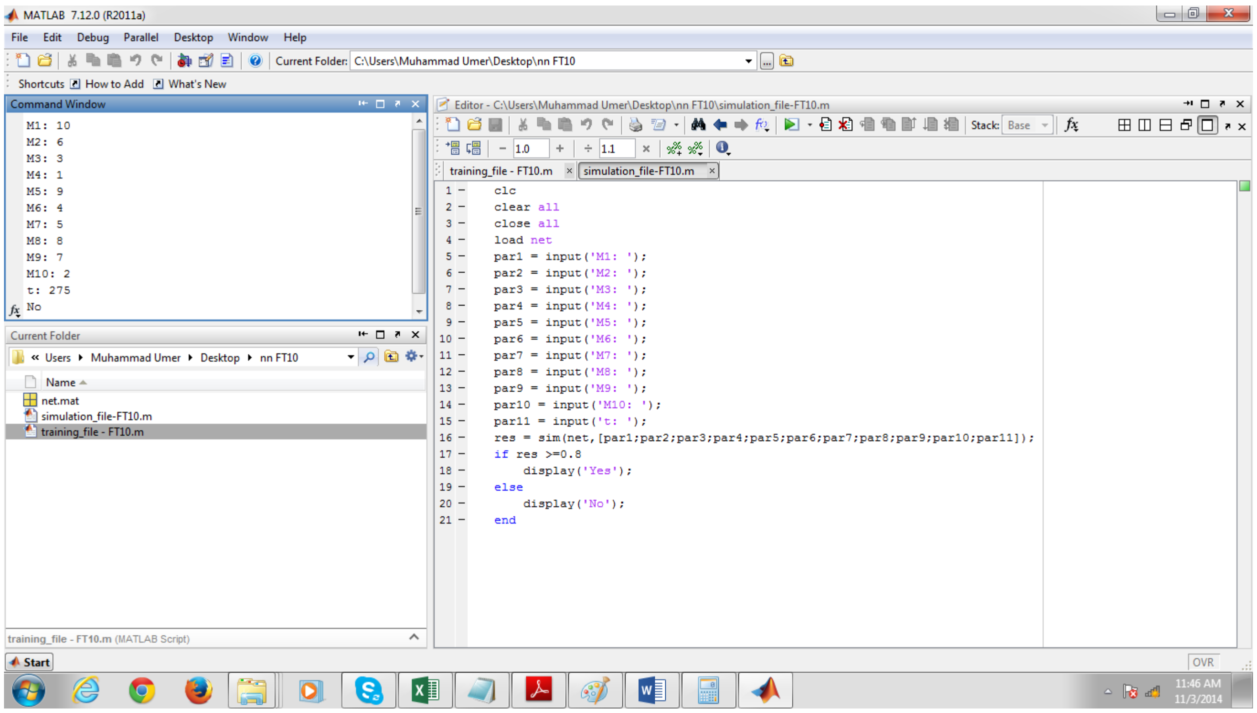

MATLAB 7.12.0 (R2011a)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\Muhammad Umer\Desktop\nn FT10
Command Window
Editor - C:\Users\Muhammad Umer\Desktop\nn FT10\simulation_file-FT10.m
training_file - FT10.m
simulation_file-FT10.m
Current Folder
net.mat
simulation_file-FT10.m
training_file - FT10.m

```
M1: 3
M2: 6
M3: 10
M4: 1
M5: 4
M6: 9
M7: 5
M8: 8
M9: 7
M10: 2
t: 395
Yes
```

```
clc
clear all
close all
load net
par1 = input('M1: ');
par2 = input('M2: ');
par3 = input('M3: ');
par4 = input('M4: ');
par5 = input('M5: ');
par6 = input('M6: ');
par7 = input('M7: ');
par8 = input('M8: ');
par9 = input('M9: ');
par10 = input('M10: ');
par11 = input('t: ');
res = sim(net,[par1;par2;par3;par4;par5;par6;par7;par8;par9;par10;par11]);
if res >=0.8
    display('Yes');
else
    display('No');
end
```

Figure 6. 10NaACO returns a YES for a likely infeasible combination

YES.



```
M1: 10
M2: 6
M3: 3
M4: 1
M5: 9
M6: 4
M7: 5
M8: 8
M9: 7
M10: 2
t: 275
No
```

```
clc
clear all
close all
load net
par1 = input('M1: ');
par2 = input('M2: ');
par3 = input('M3: ');
par4 = input('M4: ');
par5 = input('M5: ');
par6 = input('M6: ');
par7 = input('M7: ');
par8 = input('M8: ');
par9 = input('M9: ');
par10 = input('M10: ');
par11 = input('t: ');
res = sim(net,[par1;par2;par3;par4;par5;par6;par7;par8;par9;par10;par11]);
if res >=0.8
    display('Yes');
else
    display('No');
end
```

Figure 6.11 represents a sequence which is not accepted and hence the NaACO returns a NO.

6.3.3 Discussion on Results

The second problem set selected for validation of NaACO is FT10. This problem set is a genuine NP-hard solution provider as the solution space is incredibly immense. The NaACO achieves the initial convergence within ACO domain to achieve BKS. However as noticed in the results the average deviation makespan is around 14%. This deviation is largely due to the fact that the solution pendulates between extreme values for the sequences generated through ACO. This fluctuation is the core reason for this large average percentage deviation in makespan. This in itself proposes a future area for research within the domain of NaACO in general and ACO in particular so as to monitor and analyse this pendulation once the convergence is achieved for larger problem sets. This having said, still the time taken by ACO to present the solution is a great offset as compared to other techniques as already discussed in Chapter 3 of this thesis. Moreover as pointed out for FT06 problem sets the post optimal analysis in post ANN domain phase also presents an area for future research.

6.4 FT20:

The FT20 problem set is a NP-very hard problem set (the complete code for NaACO is in Appendix B). This scheduling problem involves scheduling of 20 jobs on 5 machines. The following matrix explains the machine assignment to jobs. There are 20 jobs which are to be assigned to five machines according to the following matrix:

Table 6. 9 Machine matrix for FT20

Jobs	Machines				
J1	1	2	3	4	5
J2	1	2	4	3	5
J3	2	1	3	5	4
J4	2	1	5	3	4
J5	3	2	1	4	5
J6	3	2	5	1	4
J7	2	1	3	4	5
J8	3	2	1	4	5
J9	1	4	3	2	5
J10	2	3	1	4	5
J11	2	4	1	5	3
J12	3	1	2	4	5
J13	1	3	2	4	5
J14	3	1	2	4	5
J15	1	2	5	3	4
J16	2	1	4	5	3
J17	1	3	2	4	5
J18	1	2	5	3	4
J19	2	3	1	4	5
J20	1	2	3	4	5

These machines are having separate processing times. These processing times are associated with each machine as in the following time matrix (table 6.11).

Table 6. 10Time matrix for FT20

J/M	MI	M2	M3	M4	M5
J1	29	9	49	62	44
J2	43	75	46	69	72
J3	39	91	90	45	12
J4	71	81	85	22	9
J5	26	22	14	21	72
J6	47	52	84	6	48
J7	61	46	32	32	30
J8	32	46	31	19	36
J9	76	40	85	76	26
J10	64	85	61	47	90
J11	11	78	21	36	56
J12	11	28	90	46	30
J13	85	10	74	89	33
J14	99	52	95	98	43
J15	6	61	49	53	69
J16	95	2	25	72	65
J17	37	21	13	89	55
J18	86	74	48	79	88
J19	11	69	51	89	74
J20	13	7	76	52	45

For the said problem, the methodology is to assign 20 jobs on 5 machines through ACO to obtain ψ_1 and then to validate the NaACO through neural training of the value obtained through calculation of Tau. The validation is done within the preview of the calculated results to demonstrate the accuracy of ANN. The complete NaACO code is given in Appendix B.

6.4.1 ACO Domain

The reciprocal of the heuristic function is calculated based on the greedy heuristic of minimum processing time. This implies that for this problem η = 1/ min.processing time. The values are as shown in table 6.12.

Table 6. 11Values of η for FT20

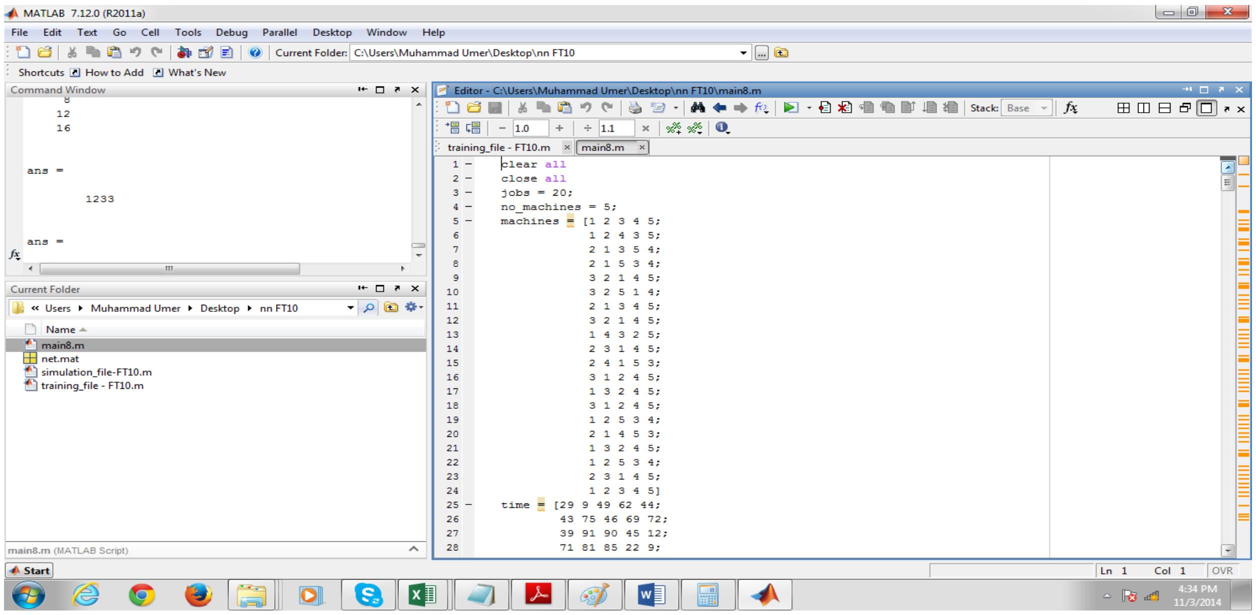
0.034483	0.111111	0.020408	0.016129	0.022727
0.023256	0.013333	0.021739	0.014493	0.013889
0.025641	0.010989	0.011111	0.022222	0.083333
0.014085	0.012346	0.011765	0.045455	0.111111
0.038462	0.045455	0.071429	0.047619	0.013889
0.021277	0.019231	0.011905	0.166667	0.020833
0.016393	0.021739	0.03125	0.03125	0.033333
0.03125	0.021739	0.032258	0.052632	0.027778
0.013158	0.025	0.011765	0.013158	0.038462
0.015625	0.011765	0.016393	0.021277	0.011111
0.090909	0.012821	0.047619	0.027778	0.017857
0.090909	0.035714	0.011111	0.021739	0.033333
0.011765	0.1	0.013514	0.011236	0.030303
0.010101	0.019231	0.010526	0.010204	0.023256
0.166667	0.016393	0.020408	0.018868	0.014493
0.010526	0.5	0.04	0.013889	0.015385
0.027027	0.047619	0.076923	0.011236	0.018182
0.011628	0.013514	0.020833	0.012658	0.011364
0.090909	0.014493	0.019608	0.011236	0.013514
0.076923	0.142857	0.013158	0.019231	0.022222

Based on these values the probabilities of each job to be assigned are calculated by incorporating the fundamental ACO equation with no pheromone upgrade to give a sequence of machines and a sequence of sorted times:

Table 6. 12Machine sorting for FT20

Sorted Machines by ACO					
J1	2	1	5	3	4
J2	1	4	3	5	2
J3	4	2	5	3	1
J4	4	3	2	1	5
J5	1	4	2	3	5
J6	1	3	4	2	5
J7	5	3	4	1	2
J8	4	1	3	5	2
J9	5	4	1	2	3
J10	4	1	2	3	5
J11	2	1	5	3	4
J12	3	1	5	4	2
J13	3	5	2	1	4
J14	5	1	2	4	3
J15	1	5	3	2	4
J16	1	4	3	5	2
J17	2	3	1	5	4
J18	5	2	3	1	4
J19	2	1	3	5	4
J20	2	1	5	4	3

The next step is the sequencing of machines on the basis of no overlapping and then spreading the machines to avoid multiprocessing. This is done by sorting the probabilities of ACO so as to assign the jobs to the machine having the least processing time. If there is a tie then the priority goes to the machine having the greater probability. The code is programmed in MATLAB and shows that ACO was able to achieve the minimum makespan of 1166 and an average makespan of 1233 (figure 6.12) with an error of 5.5%.


MATLAB 7.12.0 (R2011a)
File Edit Text Go Cell Tools Debug Parallel Desktop Window Help
Current Folder: C:\Users\Muhammad Umer\Desktop\nn FT10
Shortcuts How to Add What's New
Command Window
Current Folder
Users Muhammad Umer Desktop nn FT10
Name
main8.m
net.mat
simulation_file-FT10.m
training_file - FT10.m
main8.m (MATLAB Script)
Editor - C:\Users\Muhammad Umer\Desktop\nn FT10\main8.m
training_file - FT10.m
main8.m
Stack: Base
Start
Ln 1 Col 1 OVR
4:34 PM
11/3/2014

```
     8
    12
    16


ans =

        1233


ans =
```

```
clear all
close all
jobs = 20;
no_machines = 5;
machines = [1 2 3 4 5;
            1 2 4 3 5;
            2 1 3 5 4;
            2 1 5 3 4;
            3 2 1 4 5;
            3 2 5 1 4;
            2 1 3 4 5;
            3 2 1 4 5;
            1 4 3 2 5;
            2 3 1 4 5;
            2 4 1 5 3;
            3 1 2 4 5;
            1 3 2 4 5;
            3 1 2 4 5;
            1 2 5 3 4;
            2 1 4 5 3;
            1 3 2 4 5;
            1 2 5 3 4;
            2 3 1 4 5;
            1 2 3 4 5]
time = [29 9 49 62 44;
        43 75 46 69 72;
        39 91 90 45 12;
        71 81 85 22 9;
```

Figure 6. 11Solution through ACO in MATLAB

Table 6.14 presents the summary of ACO inspired solution. The discussion on the outcomes is covered in sub section 6.4.3.

Table 6. 13Summary for FT20

Parameters	BKS	Proposed Algorithm
Make span (sec)	1165	**1166**
Avg Make span (sec)		**1233**
% Deviation (Avg make span)		5.5%

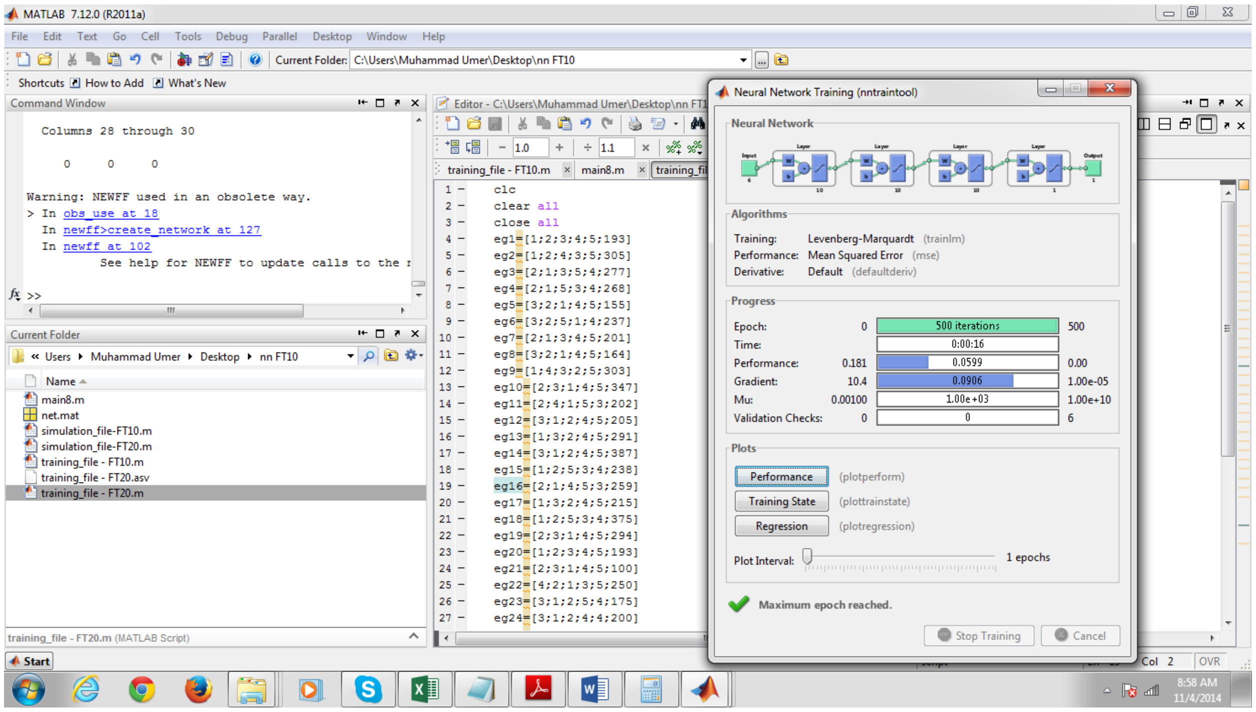
6.4.2 ANN Domain

With this sequence the value for $\psi_1$is obtained. For the validation of NaACO the model is trained on inputs, tau values and outputs. The training is done to verify the results within the known domain of answers. Firstly the system is customized to give tau values and the average tau value as in table 6.15.

Table 6. 14tau values for ANN training of the problem set.

		Tau
J1	2 1 5 3 4	0.00518135
J2	1 4 3 5 2	0.00327869
J3	4 2 5 3 1	0.00361011
J4	4 3 2 1 5	0.00373134
J5	1 4 2 3 5	0.00645161
J6	1 3 4 2 5	0.00421941
J7	5 3 4 1 2	0.00497512
J8	4 1 3 5 2	0.00609756
J9	5 4 1 2 3	0.00330033
J10	4 1 2 3 5	0.00288184
J11	2 1 5 3 4	0.0049505
J12	3 1 5 4 2	0.00487805
J13	3 5 2 1 4	0.00343643
J14	5 1 2 4 3	0.00258398
J15	1 5 3 2 4	0.00420168
J16	1 4 3 5 2	0.003861
J17	2 3 1 5 4	0.00465116
J18	5 2 3 1 4	0.00266667
J19	2 1 3 5 4	0.00340136
J20	2 1 5 4 3	0.00518135
	Avg=	0.004177

This value of tau along with the inputs of machine sequence, and make-span for each combination is utilized to constitute an ANN which is trained on the current system. This training is accomplished through supervised learning of inputs versus the outputs. At the termination of training the simulations are carried out to ascertain the validity of the model within the defined and known answers.


MATLAB 7.12.0 (R2011a)
File Edit Text Go Cell Tools Debug Parallel Desktop Window Help
Current Folder: C:\Users\Muhammad Umer\Desktop\nn FT10
Shortcuts How to Add What's New
Command Window
Current Folder
Users ▸ Muhammad Umer ▸ Desktop ▸ nn FT10
Name
Editor - C:\Users\Muhammad Umer\Desktop\nn FT1
training_file - FT10.m
main8.m
Neural Network Training (nntraintool)
Neural Network
Input
Layer
Output
Plots
Performance
Training State
Regression
Stop Training
Cancel
Start
Col 2
OVR
8:58 AM
11/4/2014

```
  Columns 28 through 30

     0     0     0

Warning: NEWFF used in an obsolete way.
> In obs_use at 18
  In newff>create_network at 127
  In newff at 102
          See help for NEWFF to update calls to the r
>>
```

main8.m
net.mat
simulation_file-FT10.m
simulation_file-FT20.m
training_file - FT10.m
training_file - FT20.asv
training_file - FT20.m

training_file - FT20.m (MATLAB Script)

```
clc
clear all
close all
eg1=[1;2;3;4;5;193]
eg2=[1;2;4;3;5;305]
eg3=[2;1;3;5;4;277]
eg4=[2;1;5;3;4;268]
eg5=[3;2;1;4;5;155]
eg6=[3;2;5;1;4;237]
eg7=[2;1;3;4;5;201]
eg8=[3;2;1;4;5;164]
eg9=[1;4;3;2;5;303]
eg10=[2;3;1;4;5;347]
eg11=[2;4;1;5;3;202]
eg12=[3;1;2;4;5;205]
eg13=[1;3;2;4;5;291]
eg14=[3;1;2;4;5;387]
eg15=[1;2;5;3;4;238]
eg16=[2;1;4;5;3;259]
eg17=[1;3;2;4;5;215]
eg18=[1;2;5;3;4;375]
eg19=[2;3;1;4;5;294]
eg20=[1;2;3;4;5;193]
eg21=[2;3;1;4;5;100]
eg22=[4;2;1;3;5;250]
eg23=[3;1;2;5;4;175]
eg24=[3;1;2;4;4;200]
```

Algorithms

Training:	Levenberg-Marquardt	(trainlm)
Performance:	Mean Squared Error	(mse)
Derivative:	Default	(defaultderiv)

Progress

Epoch:	0	500 iterations	500
Time:		0:00:16	
Performance:	0.181	0.0599	0.00
Gradient:	10.4	0.0906	1.00e-05
Mu:	0.00100	1.00e+03	1.00e+10
Validation Checks:	0	0	6

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

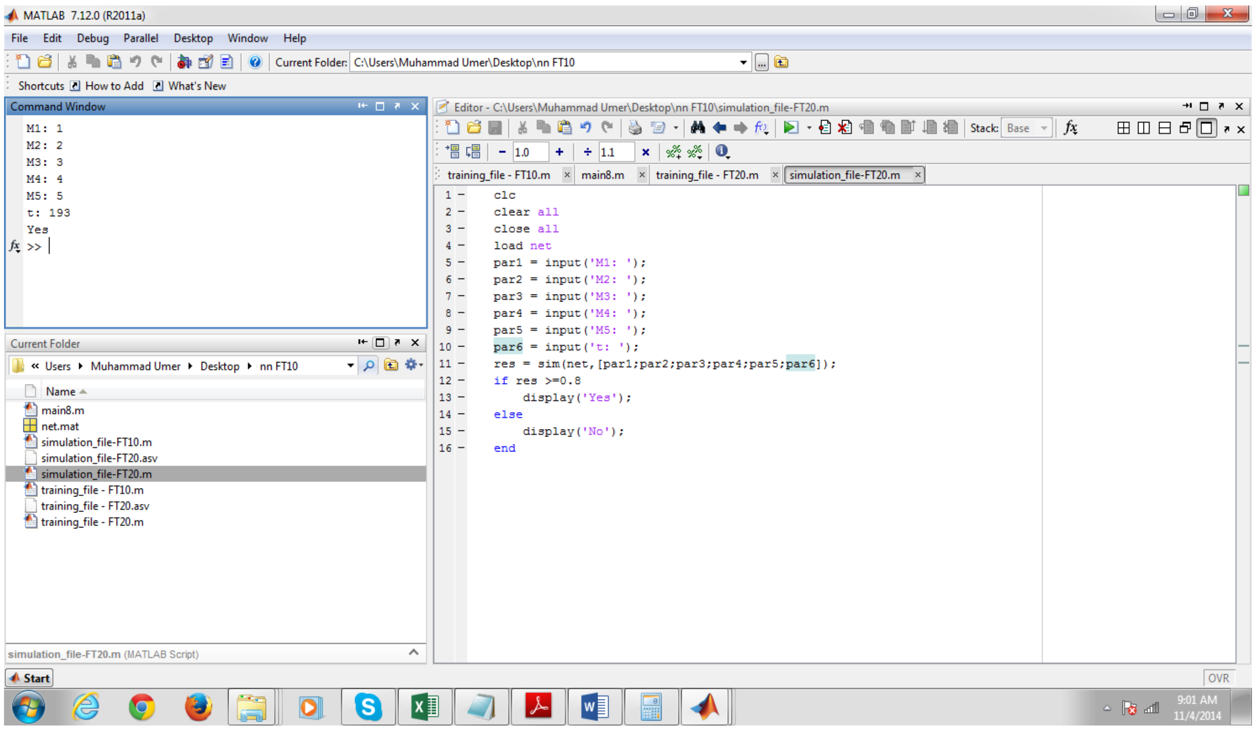
Plot Interval: 1 epochs

Maximum epoch reached.

Figure 6. 12Training of ANN for NaACO

After training of the ANN for NaACO the validation is ascertained through testing of NaACO by prompting it for a correct i.e. feasible or an incorrect i.e. potentially infeasible combination. Figure 6.14 depicts a correct and an incorrect combination of the machine sequence and times to show the validity of the model within the specified domains.

.

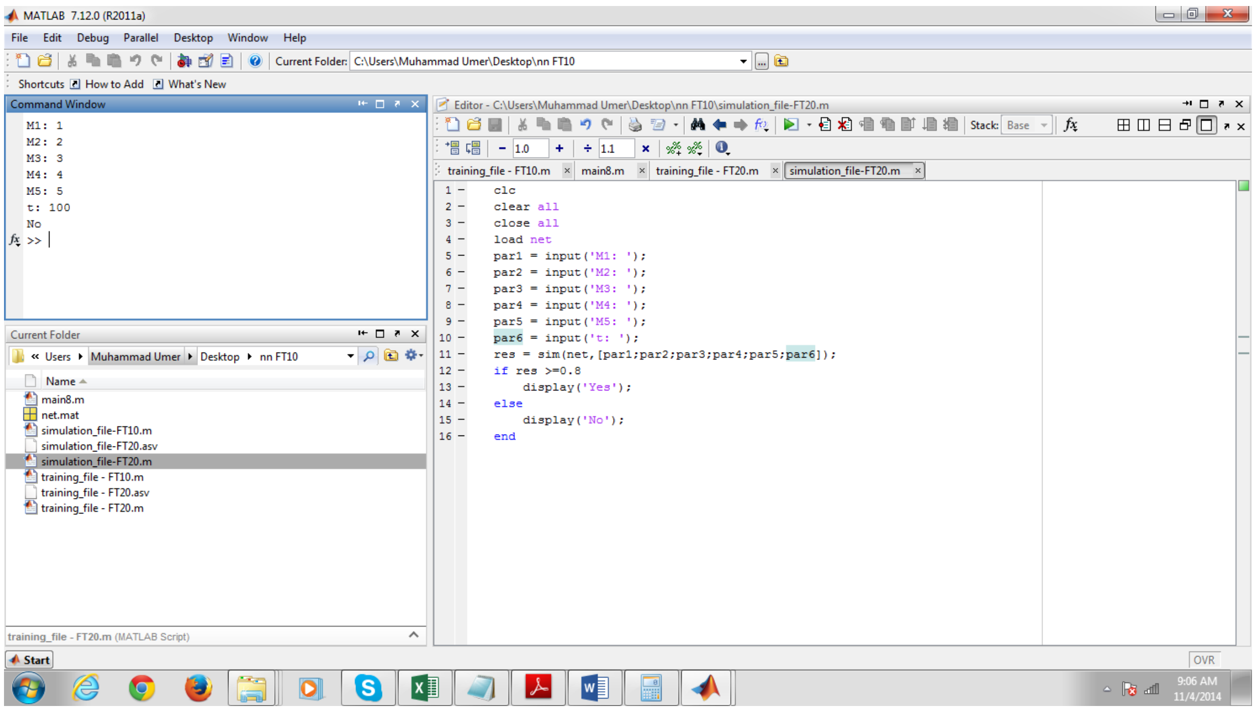


```
M1: 1
M2: 2
M3: 3
M4: 4
M5: 5
t: 193
Yes
>>
```

```
clc
clear all
close all
load net
par1 = input('M1: ');
par2 = input('M2: ');
par3 = input('M3: ');
par4 = input('M4: ');
par5 = input('M5: ');
par6 = input('t: ');
res = sim(net,[par1;par2;par3;par4;par5;par6]);
if res >=0.8
    display('Yes');
else
    display('No');
end
```

Figure 6. 13Validation of ANN domain of NaACO

Figure 6.15 represents a sequence which is not accepted and hence the NaACO returns a NO.



```
M1: 1
M2: 2
M3: 3
M4: 4
M5: 5
t: 100
No
>>
```

```
clc
clear all
close all
load net
par1 = input('M1: ');
par2 = input('M2: ');
par3 = input('M3: ');
par4 = input('M4: ');
par5 = input('M5: ');
par6 = input('t: ');
res = sim(net,[par1;par2;par3;par4;par5;par6]);
if res >=0.8
    display('Yes');
else
    display('No');
end
```

Figure 6. 14Validation of ANN domain of NaACO

6.4.3 Discussion on Results

To summarize the validation process for FT20 NaACO has been able to converge toward the BKS though ACO and the ANN domain has also been able to forecast the situations within the known set of answers. The deviation average % deviation in make span for FT20 is 5.5% which is significantly lower than that of FT10. One of the reasons for this observation may be due to the larger set of work stations which are potentially available for FT10 than that of FT20. Even so in both the cases the algorithm has been able to achieve the BKS. This aspect can also be evaluated as a potential future area of research in which the comparative analysis of FT10 and FT20 can be done on the basis of % average deviation within the domain of NaACO and further offset algorithms can be developed for futuristic investigations.

6.5 Discussion on the validation of NaACO (FT06, FT10 and FT20)

In order to converge on the outputs of the whole validation process of NaACO following discussion and finer points are put forward:

- Firstly the NaACO approach was validated on FT06 problem set. This problem set is a medium to low level validation problem set. When the NaACO methodology was implemented on this problem set, the ACO component of NaACO reached remarkable results by both finding the BKS and also by converging on the solution in the minimum of times, i.e. 0.48 seconds. This achievement reflects on the fast convergence capability of ACO. Moreover, when the ANN component was launched the NaACO has returned expected values and answers which commensurate with the desired results. The training of ANN component for FT06 takes less of time because of the fact that the combinations generated out of the 6x6 matrix are quickly learned by the supervised ANN. This makes a real life problem which looks similar to FT06 problem fairly easy to handle by NaACO. In real situations as well a FJSSP with six work stations and six job timings can be found more frequently so it can be concluded with confidence that NaACO should be able to handle a 6x6 problem set with accuracy and with fast convergence capabilities. The forecasted domain has also responded with accuracy and preciseness. This shall also encourage the future researchers to tackle the NaACO approach by *unsupervised learning* methodology.

- The second validation problem set chosen in this research was FT10. This problem set comes in the np-hard category of problems. In this problem set there are 10 jobs which are to be scheduled on 10 machines with allocated processing times. Once this problem is fed to NaACO, the ACO domain finds the BKS to FT10. The point to note over here is that while doing so the average % deviation comes out to be around 14%. This deviation is generated because of the oscillating results once the algorithms is in the process of converging on the BKS. This aspect in itself presents an area of further investigation so as to investigate this behavior of ACO component of NaACO and to study the effects of this tendency on the convergence capabilities of ACO. The ANN component of NaACO handles the supervised learning and simulation processes with considerable efficiency. Although it must be noted here that the scenario building and training of ANN in FT10 takes considerable time and deliberation as compared to the ANN training which was done for FT06. This is due to a larger data set of 10x10 matrix and the possible solutions which can be generated. In this aspect it is emphasized here that the presently employed ANN for FT10 is a 3-layered ANN. The layers of ANN can be further increased to see the effects of neural learning and the possible impact of increasing the neural layers on the outputs can be taken as future research. As FT10 problem set presents a fairly complex situation of FJSSP further investigation is required on the aspect of giving an *unsupervised-neural* interface to NaACO so as to further build on the capabilities of NaACO. The feedback mechanisms can greatly influence on the forecasting capabilities of NaACO once it tries to deal with bigger problem sets. So if a feedback mechanism can be incorporated in the ANN domain of NaACO so as to train the data sets on the run time basis, this can generate interesting and very capable variants of NaACO for future research. So all in all for FT10, NaACO confirms to the results but shows fluctuations during convergence which is one of the limitations of NaACO.

- The third problem set taken was FT20. This problem set is also np-hard and comprises of scheduling 20 jobs on 5 machines with allocated processing times. The ACO domain handles the convergence with efficiency and returns the BKS results. Moreover the average % deviation from the BKS in this case is 5.5%. This value is fairly less than as observed with FT10. This insight also suggests that as the search space grows in terms of number of available food sources (nests) the ACO algorithm demonstrate the pendulum tendency and this tendency is irrespective of the number of ant (jobs) available in a scheduling environment. As observed during the validation process even if the number of jobs are doubled i.e. from 10 jobs in FT10 the jobs are increased to 20 jobs in FT20 NaACO finds the optimum solution with the minimum of % average deviations but when the number of sources are increased or doubled i.e. in FT20 there are 5 machines and in FT10 there are 10 machines the fluctuation or pendulum tendency is observed which increases the % average deviation. This area can be taken up by future researchers as an interesting point of departure for having an insight towards further evolution of ACO algorithm. The optimization paradigm is confirmed by ANN domain by FT20 and future scenarios are forecasted within the known solution space.

- Once a comparison is done on the whole validation process it can be inferred that the ACO formulation strengthens or weakens the overall formulation of NaACO as the ANN domain is directly fed by ACO domain for forecasting purposes. Moreover the neural interface through tau can be further strengthened by incorporating the variables α and β in the neural domain. These variables are used in classical ACO to give the boundary conditions to ACO for η and τ respectively. The neural treatment of these two values for further developments of variants for NaACO is another area for future consideration.

6.6 **Conclusion**

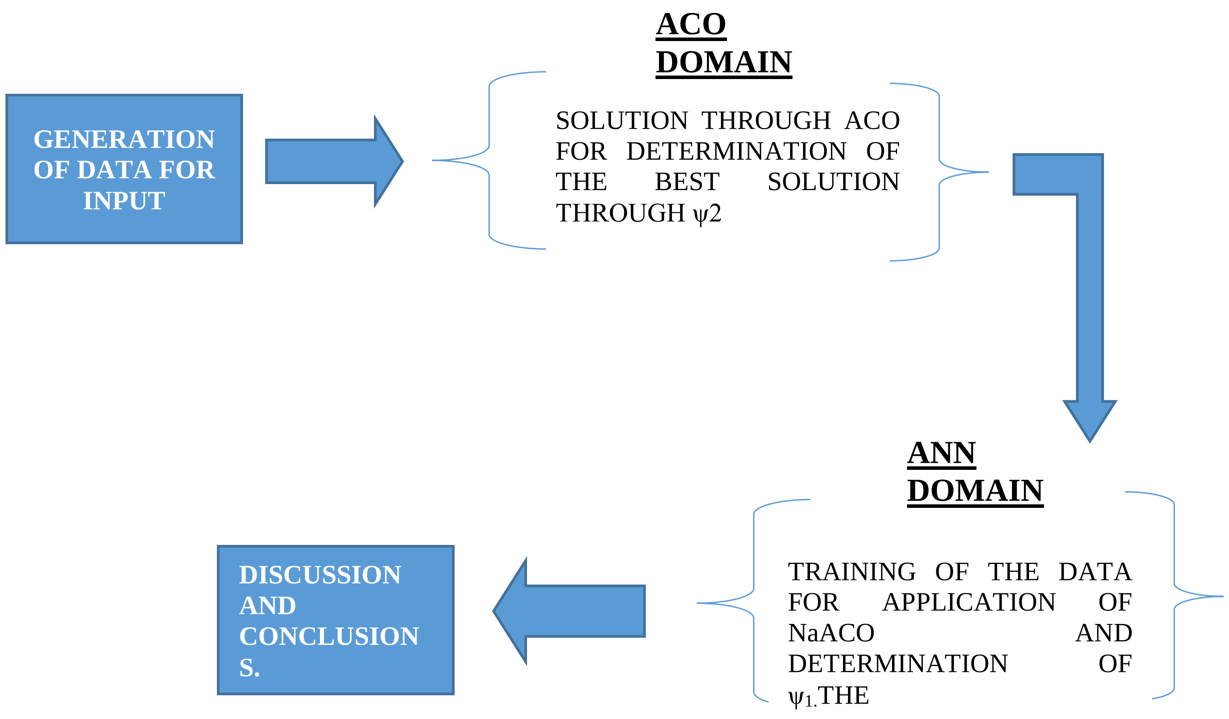
It is concluded that the application of NaACO on np hard (FT06) and np very hard problems (FT10 and FT20) depicts its functionality and response once presented with a huge problem set. The basis of NaACO is the accuracy and time taken for NaACO to forecast the feasible solutions in the context of complex and big problem sets. The purpose of this formulation and validation is to demonstrate the quick response, forecasting and future scenario building capability of NaACO.

6.7 **Chapter Summary**

In this chapter the validation of NaACO has been carried out in order to comment upon the functionality and feasibility of this technique. It is therefore concluded by this process of validation that NaACO shows coherent results which were envisaged during the conceptualization and formulation of this technique. However it is also a fact that no research is by all means a complete research i.e. without its shortcomings and follies. This shall hold true for this research as well. Further robustness approaches for NaACO can be developed in future which could be more AI intensive and could be tested on larger problem sets to give interesting results.

CHAPTER 7: APPLICATION OF NaACO ON INDUSTRY

In this chapter the application of NaACO is discussed as regards to the industrial setups. This chapter holds its own unique significance as NaACO is being tested on some industrial situations for the very first time. For this purpose a classical FJSSP problem is taken from the aviation industry and a possible application area within the domain of flowshop analysis is taken from the fan manufacturing industry. The second application area case study is



GENERATION OF DATA FOR INPUT
ACO DOMAIN
SOLUTION THROUGH ACO FOR DETERMINATION OF THE BEST SOLUTION THROUGH ψ2
ANN DOMAIN
TRAINING OF THE DATA FOR APPLICATION OF NaACO AND DETERMINATION OF ψ1.THE
DISCUSSION AND CONCLUSIONS.

Figure 7. 1 Application roadmap for NaACO

presented in Appendix C. The application of NaACO is explained by the following roadmap:

7.1 **Case Study (Application of NaACO Technique on Aviation Maintenance/ Overhaul Setup)**

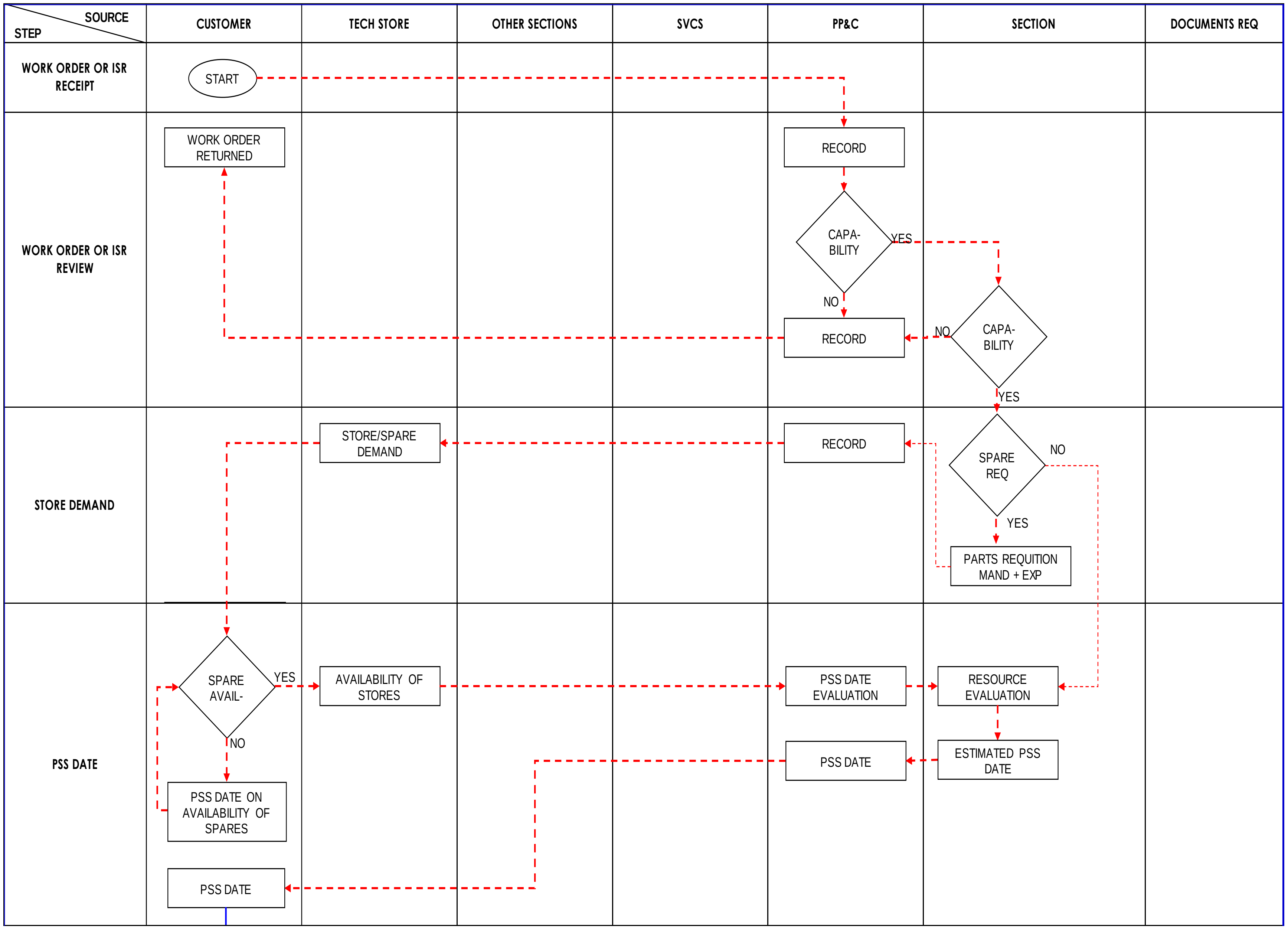
7.1.1 Background

Pakistan’s aviation maintenance industry is one of the most sophisticated and rapidly increasing industries of the country. Within this industry the small fixed wing aircrafts are most prominent as regards to the assembly, and maintenance of their various equipment and components. As per the context of this research one of the biggest engine overhauling facilities of these small fixed wing aircrafts is being selected for the application of NaACO. These aircrafts are mounted with four cylinder piston engines (IO-360-A1B6). There are various assembly lines which are in parallel for the overall assembly of these engines. Within these engine assembly lines the research has focused on the smaller components which are required in a greater number and with more frequency. A typical engine process flow for the disassembly and inspection is depicted by figure7.1.

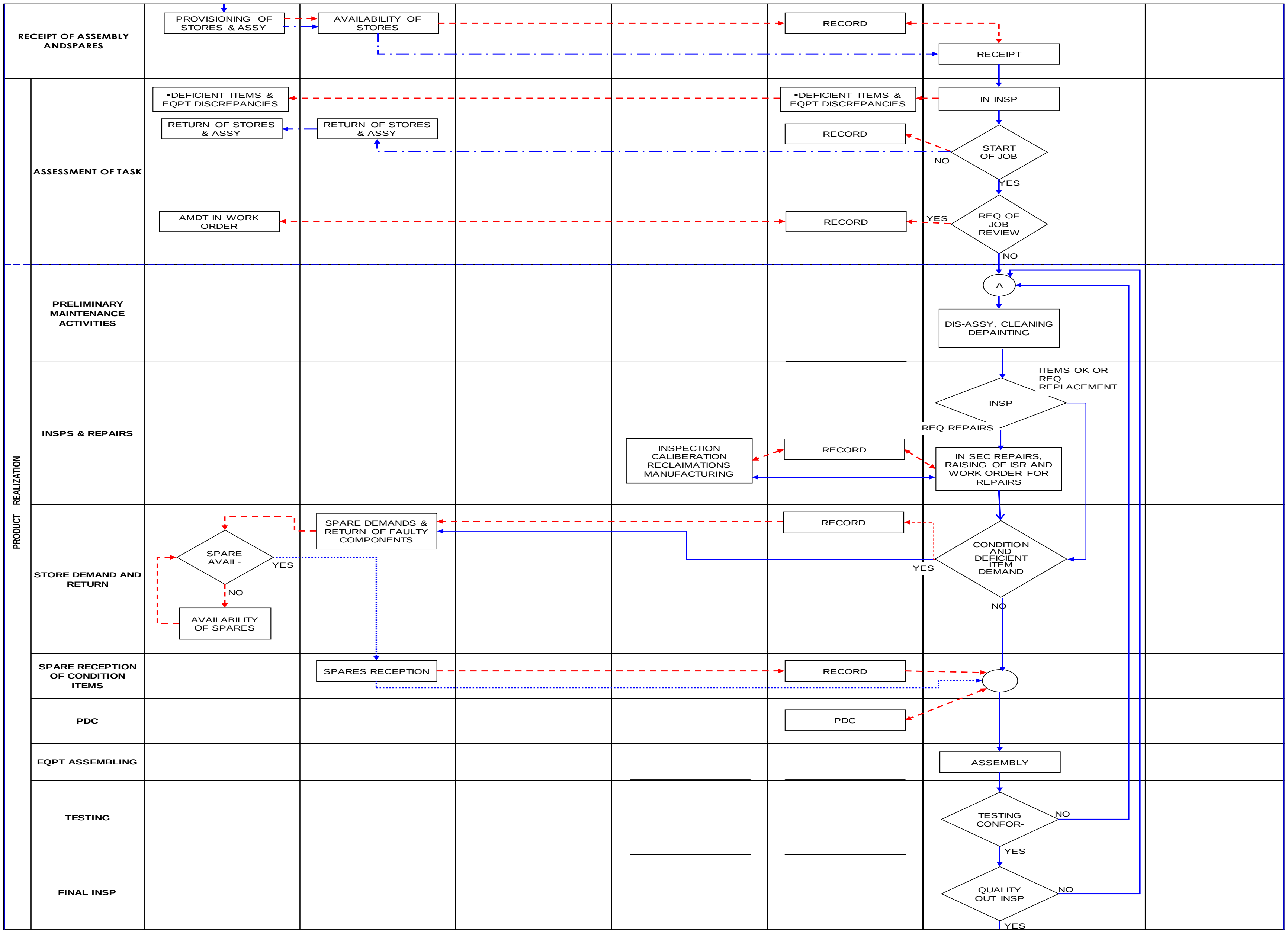
7.1.2 Model Initialization

The servicing stations of Magnetos (the ignition devices) have been picked for workstations. The ignition is termed as MAGNETO BENDIX. These magnetos are required by the fixed wing aircrafts on a regular basis, so they have a very frequent turn around. This research has focused on 500 hours inspection of these magnetos. In particular the assembly line is faced with a FJSSP scenario when the cleaning of these devices is done after the disassembly. There are ***three*** cleaning bays on the workshop floor. As per the process sheets of 500 hours inspection each cleaning bay has three operations: Chemical cleaning with Trichlorethylene, Rubbing with 00-00 embry paper and compressed air cleaning at 30 psi. These three sub processes are named as TriClean, RubClean and AirClean for model. Each sub process has a particular time which varies with the experience of worker, the efficiency of the worker and various other related factors. Now consider that there are a number N of magnetos which are to be cleaned at these three cleaning bays, each having all the three sub processes.

As per the process sheet of the operation (figure 7.1) at each cleaning bay any one of the operations has to be happening and no two operations can be performed on any bay at a given time. This restriction is due to the hazardous materials involved and also due to the intricate nature of the sub processes. Therefore, there is a situation in which the sequence of operations can be (TriClean1, RubClean2, AirClean3) or (Air Clean3, RubClean2, TriClean1) etc. Each operation is independent of each other which means that there is no precedence of the three operations.



SOURCE
STEP
CUSTOMER
TECH STORE
OTHER SECTIONS
SVCS
PP&C
SECTION
DOCUMENTS REQ
WORK ORDER OR ISR RECEIPT
START
WORK ORDER OR ISR REVIEW
WORK ORDER RETURNED
RECORD
CAPA-BILITY
YES
NO
RECORD
NO
CAPA-BILITY
YES
STORE DEMAND
STORE/SPARE DEMAND
RECORD
SPARE REQ
NO
YES
PARTS REQUITION MAND + EXP
PSS DATE
SPARE AVAIL-
YES
AVAILABILITY OF STORES
PSS DATE EVALUATION
RESOURCE EVALUATION
NO
PSS DATE ON AVAILABILITY OF SPARES
PSS DATE
ESTIMATED PSS DATE
PSS DATE



RECEIPT OF ASSEMBLY ANDSPARES
PROVISIONING OF STORES & ASSY
AVAILABILITY OF STORES
RECORD
RECEIPT
ASSESSMENT OF TASK
▪DEFICIENT ITEMS & EQPT DISCREPANCIES
▪DEFICIENT ITEMS & EQPT DISCREPANCIES
IN INSP
RETURN OF STORES & ASSY
RETURN OF STORES & ASSY
RECORD
START OF JOB
NO
YES
AMDT IN WORK ORDER
RECORD
YES
REQ OF JOB REVIEW
NO
PRODUCT REALIZATION
PRELIMINARY MAINTENANCE ACTIVITIES
A
DIS-ASSY, CLEANING DEPAINTING
INSPS & REPAIRS
ITEMS OK OR REQ REPLACEMENT
INSP
REQ REPAIRS
INSPECTION CALIBERATION RECLAIMATIONS MANUFACTURING
RECORD
IN SEC REPAIRS, RAISING OF ISR AND WORK ORDER FOR REPAIRS
STORE DEMAND AND RETURN
SPARE DEMANDS & RETURN OF FAULTY COMPONENTS
RECORD
SPARE AVAIL-
YES
NO
AVAILABILITY OF SPARES
CONDITION AND DEFICIENT ITEM DEMAND
YES
NO
SPARE RECEPTION OF CONDITION ITEMS
SPARES RECEPTION
RECORD
PDC
PDC
EQPT ASSEMBLING
ASSEMBLY
TESTING
TESTING CONFOR-
NO
YES
FINAL INSP
QUALITY OUT INSP
NO
YES

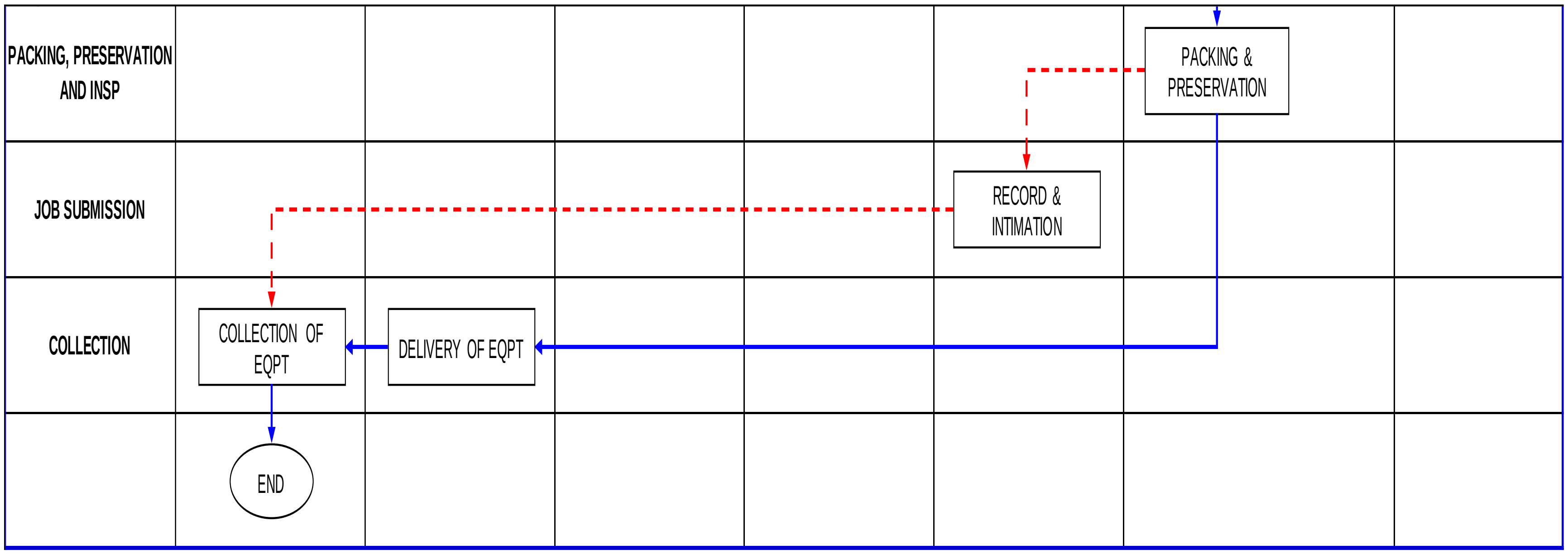


Figure 7. 2 Process Map for Magnetto 500

The facility itself works on a weekly schedule, i.e. the output is the number of Magnetos cleaned per week. We have obtained random data for the cleaning process of 10 magnetos spread along four to six weeks. Moreover we also have an initial workforce of 10 technicians to be assigned to the three bays. The data sets thus obtained for the model initialization (figure 7.1) and model implementation were as follows:

TC = TriClean; RC = RubClean; AC = AirClean

Time: In minutes.

Table 7. 1data tables for initiation of NaACO

Week1[*]

	BAY1			**BAY2**			**BAY3**		
Magnetos	**TC1**	**RC1**	**AC1**	**TC2**	**RC2**	**AC2**	**TC3**	**RC3**	**AC3**
1	30	45	24	25	30	22	40	33	30
2	25	40	22	19	24	28	27	32	45
3	35	35	27	20	23	29	27	29	33
4	25	30	31	33	44	33	25	29	24
5	30	50	25	26	29	35	20	44	28
6	30	45	28	30	50	45	33	24	40
7	29	40	40	24	29	44	26	45	33
8	33	29	33	38	36	43	44	26	23
9	29	38	40	29	44	50	20	30	23
10	30	44	45	40	33	49	25	50	40

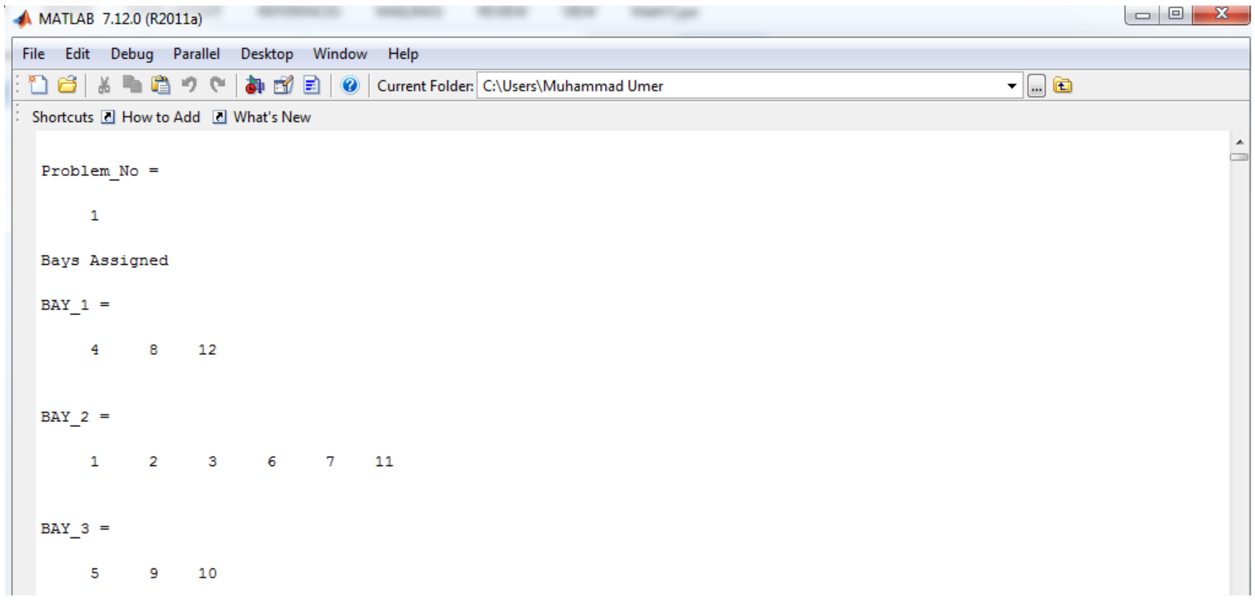
*rest can be seen in Appendix.

7.1.3 ACO Domain:

NaACO starts with the application of ACO to calculate the make span times. For this purpose the fundamental formulation is considered in which the Magnetos are the ants and the Bays are the food sources. The ACO is triggered by the heuristic of minimum processing times according to the following equation:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \; ; \text{if j} \in N_i^k \qquad \text{Equation 7.1}$$

In the above expression the value of η is the reciprocal of the heuristic function. In this case the heuristic function is the shortest processing time for each job. The values of alpha and beta are kept as 1. The initial assignments look as in figure 7.3.


MATLAB 7.12.0 (R2011a)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\Muhammad Umer
Shortcuts How to Add What's New

```
Problem_No =

     1

Bays Assigned

BAY_1 =

     4     8    12


BAY_2 =

     1     2     3     6     7    11


BAY_3 =

     5     9    10
```

Figure 7. 3 Assignment of Bays through NaACO

The same assignment is done for similar situations. In the following lines the results of five such situations (for five weeks) are presented:

<u>Results:</u>

<u>Situation. no =1</u>

Bays Assigned

```
BAY_1 =   4    8    12
BAY_2 =   1    2    3    6    7    11
BAY_3 =   5    9    10
Make_span = 139.115
```

<u>Situation no = 2</u>

Bays Assigned

```
BAY_1 = 2    8    11
BAY_2 = 5    7
BAY_3 = 1    3    4    6    9    10    12
Make_span = 183.2940
```

<u>Situation no =3</u>

Bays Assigned

```
BAY_1 = 1    4    6    8    12
BAY_2 = 10
BAY_3 = 2    3    5    7    9    11
Make_span = 147.2087
```

<u>Situation no =4</u>

Bays Assigned

```
BAY_1 = 1    2    8
BAY_2 = 3    6    9    10    11
BAY_3 = 4    5    7    12
```

Workforce Assigned

```
Make_span = 117.1107
```

Situation no 5

Bays Assigned

BAY_1 = 3 4 6 7 8 9

BAY_2 = 1 2 5 10 11 12

BAY_3 = Empty matrix: 1-by-0

Make_span = 173.2667

In the above results the make span time is the idealistic time which is being calculated by the application of ACO, thus formulating$\boldsymbol{\psi_2}$. With this initial allocation and loading of bays it is to be decided that how many Magnetos should the workshop entertain for the seventh week?

7.1.4 ANN Domain:

The value of $\boldsymbol{\psi_2}$shall also be used as one of the reference outputs once we incorporate the NaACO methodology to answer the above mentioned questions. The ideal value can be based upon by looking at the trend of $\boldsymbol{\psi_2}$throughout the problem sets.An average value can also be taken as a bench mark and in addition to that upper and lower bounds can also be developed to check for the sensitivity of this parameter. For ease of use the average estimation of$\boldsymbol{\psi_2}$ is taken as approx. 150 minutes (the average of the five weeks). Never the less this value should be compared with the existing process sheets and work study already carried out to assume based on logical approximations.

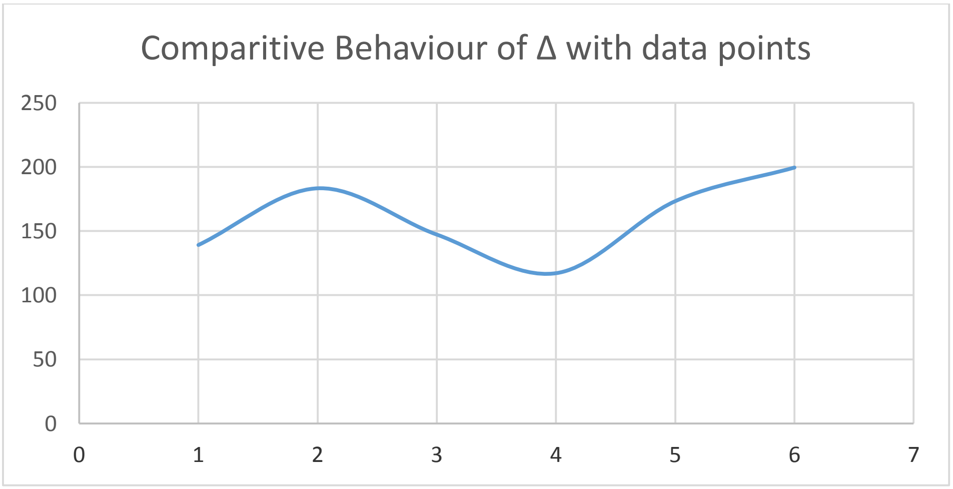


Figure 7. 4Comparative Behavior of $\boldsymbol{\psi_2}$ with Data Points

The corrective factor τ and its associated inputs are utilized (to formulate a strategy in which the system is trained through inputs in this case the three cleaning times i.e. TriClean, RubClean and AirClean) with the ideal output $\boldsymbol{\psi_2}$ and the corrective factor τ. As indicated in chapter 3, a good heuristic practice is to set the initializing pheromone trails to a unit marginally greater to the quantity of pheromone accumulated by ants in a single redundancy. A rough estimation of this unit can be achieved by setting $\forall(\boldsymbol{i},\boldsymbol{j}), \boldsymbol{\tau_{ij}} = \boldsymbol{\tau_o} = \boldsymbol{m}/\boldsymbol{C^{nn}}$, where m represents the amount of ants (in our case the number of jobs) and $\boldsymbol{C^{nn}}$ represents the distance of. As a first step the values of $\boldsymbol{\tau_o}$ is calculated for this problem and the upper and lower bounds are defined in comparison with the number of Magnetos. These values are problem specific and will vary with each problem set.

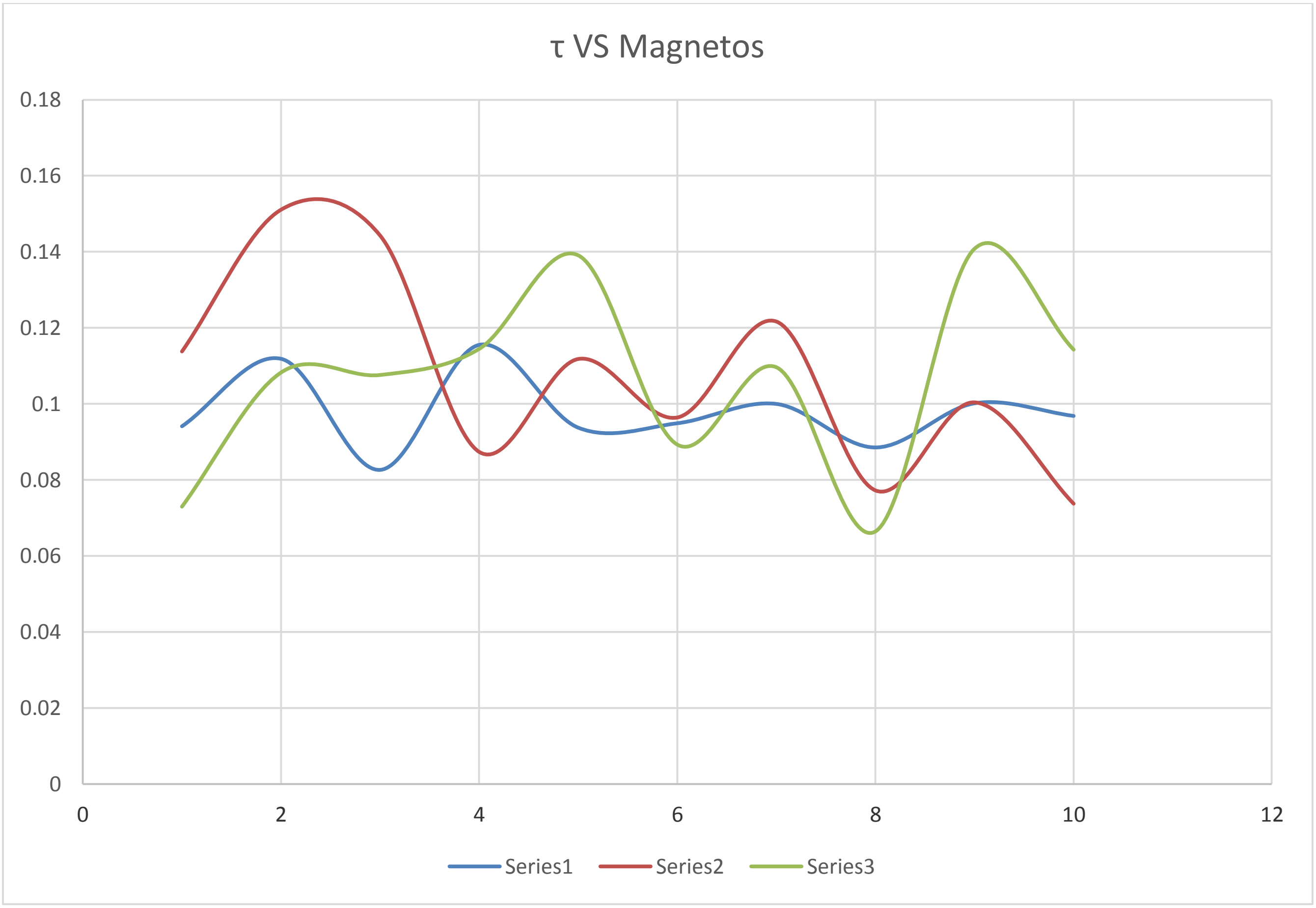
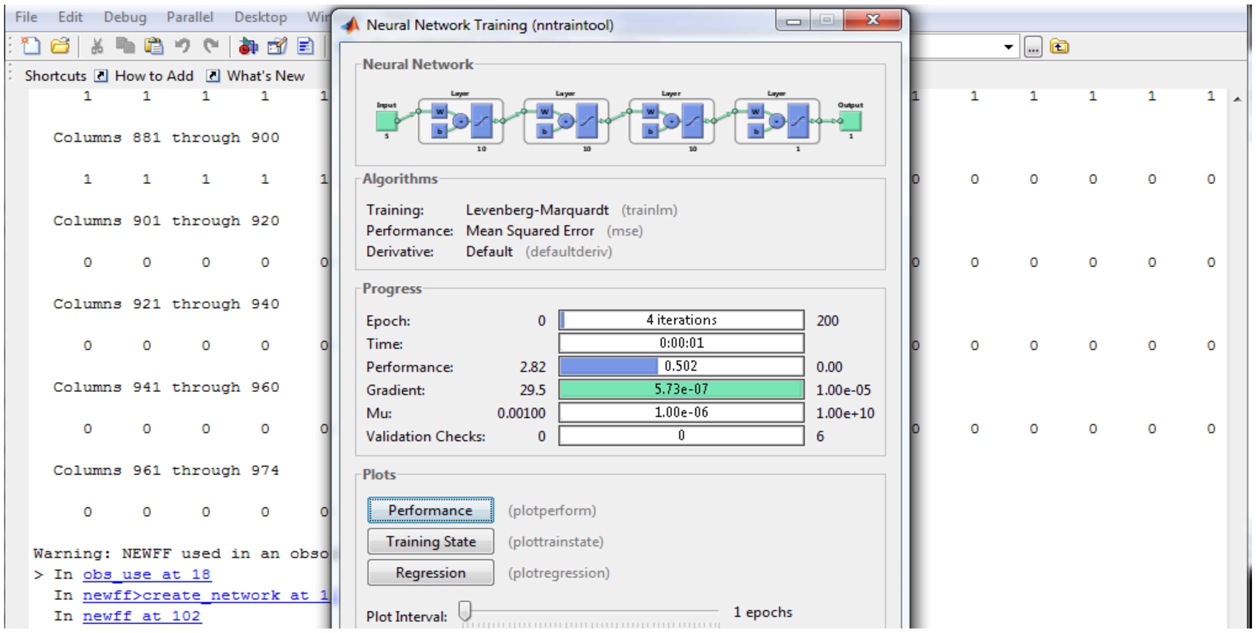


Figure 7. 5τVS Magnetos

Now the ANN is trained with the inputs (TriClean, RubClean,AirClean, number of Magnetos, workers assigned) and the outputs ($\boldsymbol{\psi_2}$ and τ).


File Edit Debug Parallel Desktop
Shortcuts How to Add What's New
Neural Network Training (nntraintool)
Neural Network
Input
Layer
Output
Performance
Training State
Regression

```
     1     1     1     1
  Columns 881 through 900
     1     1     1     1
  Columns 901 through 920
     0     0     0     0
  Columns 921 through 940
     0     0     0     0
  Columns 941 through 960
     0     0     0     0
  Columns 961 through 974
     0     0     0     0
Warning: NEWFF used in an obso
> In obs_use at 18
  In newff>create_network at 1
  In newff at 102
```

Algorithms
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)

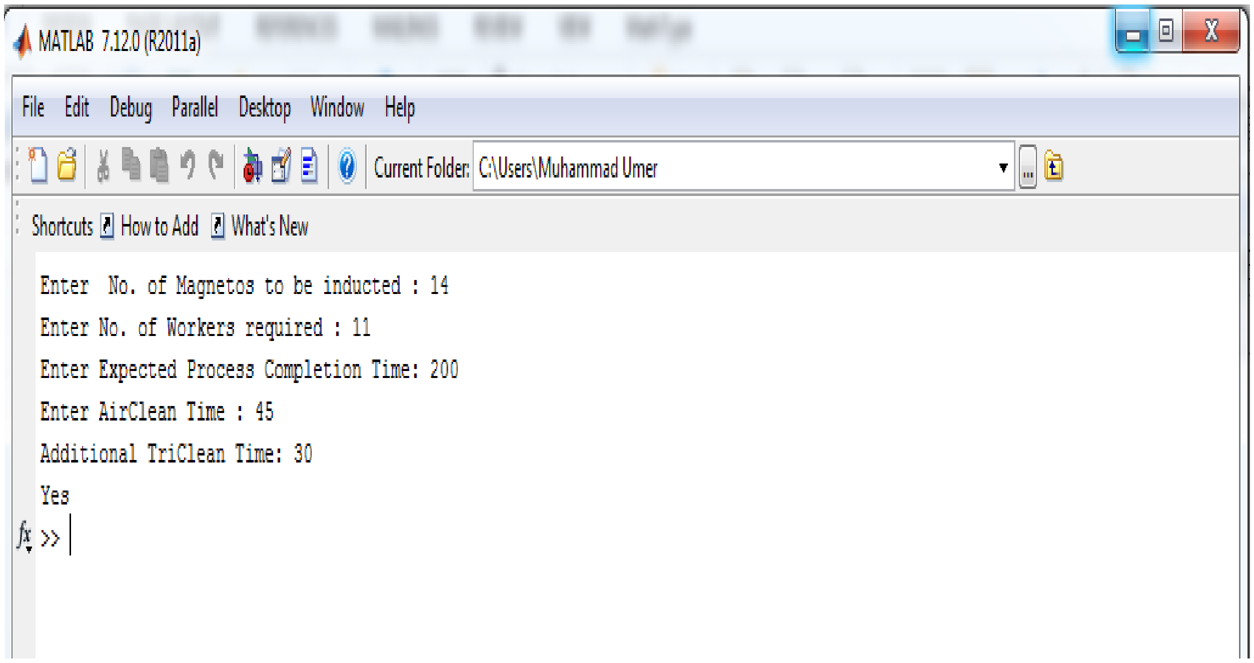
Progress

Epoch:	0	4 iterations	200
Time:		0:00:01	
Performance:	2.82	0.502	0.00
Gradient:	29.5	5.73e-07	1.00e-05
Mu:	0.00100	1.00e-06	1.00e+10
Validation Checks:	0	0	6

Plots
(plotperform)
(plottrainstate)
(plotregression)
Plot Interval: 1 epochs

Figure 7. 6 Simulation of the problem through NaACO

After the training of the ANN with [0,1] target values, NaACO is now in a position to check for the number of Magnetos which can be inducted in the seventh week. As discussed earlier; the approach is a combitorial approach coined as NaACO.


MATLAB 7.12.0 (R2011a)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\Muhammad Umer
Shortcuts How to Add What's New

```
Enter  No. of Magnetos to be inducted : 14
Enter No. of Workers required : 11
Enter Expected Process Completion Time: 200
Enter AirClean Time : 45
Additional TriClean Time: 30
Yes
>>
```

Figure 7. 7 Comitorial Dialogue Window for NaACO

The expected process completion time is $\boldsymbol{\psi_1}$ and has to be greater than $\boldsymbol{\psi_2}$ to give a realistic result i.e "YES". As a result this case now has the minimization objective function which was defined in chapter 5 as:

$$\boldsymbol{\psi}\ (\boldsymbol{\sigma}) = \boldsymbol{\psi_1} - \boldsymbol{\psi_2} \qquad \text{Equation 7.2}$$

So, in this case the value will be $\boldsymbol{\psi(\sigma) = 200 - 150 = 50 minutes.}$ This process is not a linear approximation as the inputs and the outputs have a neural based relationship which cannot be clearly defined. The results confirm the initial statements that the optimality will be nested between the ideal and worst results. This objective function can be iteratively minimizing in order to further decrease the forecasted value as a combination and to gain further efficiency for this operation.

7.2 Discussion and Conclusion

In this case the concept of NaACO is applied to an industrial setup. The setup being chosen is of a high tech aviation maintenance workshop in which the piston engines are overhauled. Within this setup there is a process of magneto overhaul and cleaning. This component is part of the ignition system for this engine. The methodology of application of NaACO involves observing the on ground situation and comparing it with the process sheets. NaACO is applied in two phases as given in figure 7.1. The ACO phase returns the ideal value or the best fit solution. The ANN domain caters for the forecasting for the completion time as a combination of various inputs. The NaACO returns the fitness of the forecasted combination as a YES or a NO. Here it is emphasized that this combination is a realistic estimate and can be further reduced based on number of associated factors. This implementation of NaACO on a live case demonstrates the capability of this model to handle real time industrial situations. This model is by no means complete for an industrial usage as it lacks the user friendliness aspects. If these aspects are undertaken as a future area then at graduate levels this application area can serve as a panacea for M.S level research works.

CHAPTER 8: CONCLUSION AND RECOMMENDATIONS

8.1 Conclusions

The findings and contributions of this research both in academic body of knowledge and applied domains are summarized as follows:

- **A Reality Based Solution has been produced**:
  The importance of handling scheduling problems is self-evidently vivid in the manufacturing industry of countries like Pakistan. It is significantly due to the fact that the supply of basic infrastructure and resources including but not limited to the energy resource (electricity, natural gas etc.) to the industry is piece meal. Such nascent conditions to run the industry pronounce two basic problems. One: The industry will not survive due to the financial and economic implications. Two: Even if the industry survives (as it is doing so right now) the demand from the end user shall spike at various instances and the industry will fall short to coupe with that demand due to its internal process malfunctions; and perhaps one of the most significant shortfall is the phenomenon of *Choking* of assembly lines, process flows due to the lack of handling expertise.

- **An Intelligent Solution to Scheduling Problems has been developed**: Although in some cases redundancy of assembly lines, process flows can be posed as a solution as well (as evident in our second case study), in situations where we already have a redundant system we still require that system to be addressed in terms of efficiency increase, cost reduction, manpower allocation, energy savings, and layout improvements. The domain of scheduling itself incorporates a sense of organizing the assembly lines for optimization, but when redundancy is combined with the scheduling domain a new form of constrained problem is evolved which has the essence of scheduling and the features of repetitiveness.

- **An Intelligent Combitorial Algorithm has been formulated:** Algorithms are the building blocks of any solution finding process. The nature of the problem itself determines which building blocks to use. It is quite possible that the solution of the problem is convergence friendly and if we use an algorithm which in itself is divergent prone, this combination shall never suffice to give us an efficient mechanism. Likewise, algorithms develop solutions in the order of priority as discussed in detail in chapter no.1. The solutions thus developed should be effective ones and the algorithm developing it should be an efficient one as well. For NP-complete problems it is often the case that we are looking for a good solution rather than a correct one. In their true nature every algorithm which finds a solution is good enough, but we rate algorithms not only because of their accurate solution finding capabilities but also because of their speed of convergence and more importantly because of their ease of use. By ease of use we mean that what inputs are required to execute that algorithms and also how quickly that algorithm can adapt to change in the original problem. The ability of an algorithm to do these things makes it the first choice for anybody in the applied sciences field. In everyday life as well we see a dynamic environment which changes due to several inputs. Artificial environments are unable to grip and absorb every detail and change, but with this limitation if an algorithms succeeds in doing a mere percentage of reality, it stands out against its counterparts.

- **A Heuristic-Neural Hybrid Technique to Tackle FJSSP has been introduced**: The job shop scheduling problems are one of the hardest problems to solve. As mentioned in chapter 2, one of the earliest attempts to solve such a problem was through branch and bound algorithm. The use of dynamic programming is also considered to be one of the initial efforts to handle such problems. As a matter of fact, all these techniques actually require a very extensive knowledge of operations research. In -fact the roots of these problems is in the development of concepts of linear programming, non-linear programming and the development of constraints. The objective function has to be defined and the best suited value of the decision variables is developed. So, it is also an iterative process. Often in daily life or in the factory workshop floor we don't have enough of the leverages to construct, validate and then use such an approach, moreover as the manufacturing industry is growing the machines are becoming efficient and the workshop floors are facing more problems to handle scheduling cases. The very nature of flexibility as deemed necessary to compete in the coming market, is creating an environment which grows with the demand of the customers. The concept of FJSSP revolves around the task and thus is not rigidly limiting the manufacturing environments.

- **Flexible and Customized Quantitative Model has been proposed:** The flexibility of a system enables the users and the people responsible to adjust to the present conditions. The flexibility also has its drawbacks. The foremost is that it puts more emphasis on the planning and the forecasting phase. Moreover, in manufacturing the flow of inventory and work in process inventory has to be lean. This situation brings about the utility of heuristic based techniques which are essential for the early development and execution of a structured approach towards this unstructured setup. So in a way it *structures* flexibility. The heuristic information which one provides has to be self-grown or most suited for the ground realities. If we look at the essence of greedy algorithms we find that they are based on the need generation mechanism. Often in industries across the boards we have different needs and even within the industry we have situations in which our priorities change. If we take example of Pakistani industry, the basic requirement or need is power/electricity/etc. This need is potent enough to present and appear as a fundamental heuristic for running of any type of a heuristic based optimization model. The offshoot of this is the efficiency gain within the industry itself.

- **ANN for Post Conditioning of a Meta Heuristic has been used:** Neural networks are adaptive to the changing inputs with respect to the outputs. In fact a biological neuron can handle around 10,000 inputs at a single time. As far as the artificial neurons are concerned, the adaptation depends upon their learning capabilities s and the type of algorithms used. If the network is well designed and if the algorithm is very over convergent or on the other hand takes time, then the advantage of having a neural end gets diminished. A useful method for the neural learning rate is the Bold Driver Algorithm. This technique takes into account the relative error values in response to the input and the outputs. The Bold Driver takes the value of error to the upper slope edge of the function and that is the point where the step size reduction takes place. In order to incorporate the network fully one requires a preconditioning methodology. The learning rate of the network greatly depends on this particular aspect. If the associated aspect and algorithm is not strong enough in the preconditioning phase the neural network will never converge in assurance towards a decision. ACO in this respect has proved to be an excellent pre conditioner to ANN both in supervised and unsupervised neural learnings. It is the convergent property of ACO which enables ANN to learn quickly and definitively. This aspect greatly improves the decision making potential of ANN assisted models. In our case as depicted in the mathematical formulation of the problem, once the ANN model for our problem is built, we preconditioned it through the introduction of Δ which was a direct output from the application of ACO.

- **Amalgamation of Pheromone Up gradation Variable (Tau) with ANN has been presented:** Moreover, the concept of inclusion of τ as an input to ANN enabled us to make use of the parameters at hand coming directly from the ACO mathematical formulations. ACO is much better than other preconditioning algorithms such as annealing such that it follows converge than search approach, whereas annealing and other associated methods follow search than converge methodology. The biggest disadvantage of ACO one might think is that it can over converge and fall into local optima, but as we have seen in our case, there are situations in which we require an algorithm to first find something for us and then refine it rather than doing the opposite. We require definite and quick answers rather the accurate ones.

- **The utility of Swarm Intelligence meta-heuristics through the amalgamation of three concepts i.e. Algorithms, Artificial Neural Networks in the application area of scheduling** have been further extended. As discussed in chapter no.1 algorithms are most potently used as a technology, a technology which is even having a greater memory map than computers, we combine it with the Artificial Intelligence of Neural

Networks and we can have a potent enough mechanism of handling complex problems. As elaborated in chapter 2 optimization technique brings about the on ground aspects to the problem. Optimization has to be intelligent and convergent to gain efficiency of the solution and for that the aspects of learning from the nature is the best resource. As a result swarm intelligence emerges as a very deliberate technique to handle the logical part of the questions and problems. In chapter 3 we have converged on the development of the mathematical model for our proposed technique and the introduction of ANNs to make the solution finding into a smart process. This process is dove tailed with the most convergent friendly of the swarm intelligence techniques, the ACO.

- **Extra-Ordinary Results by ACO have been obtained**: In order to ascertain the convergence capabilities we have demonstrated the results and the convergence times through the application of ACO on a set of 100 problems.These results were sure enough for us to conclude that ACO is the most efficient technique to handle convergent prone problems. With this aspect in mind we then combined this particular property of ACO with ANNs to generate an intelligent real time model. As discussed in chapter 3 we made use of the corrective factor of ACO i.e. tau (τ) to grasp the key parameters required for the training of ANN enabled model along with the ACO converged result. This technique was applied on various problem sets and in chapter 6 we have practically demonstrated the usage of this technique on two variant industries, one is a high tech industry of aviation maintenance in which the demand is fluctuating but whenever it comes the process lines are heavily booked. The other application area was the ceiling fan manufacturing industry of Pakistan. This industry by its nature is season dependent. Once the season is approaching the process lines get choked. In such a situation, it is pertinent enough to create a redundancy scenario. This shall ensure flow and fast track the manufacturing process.

- **Novel Formulation of NaACO:** Although there have been various attempts to tackle the issue of redundant scheduling through hill climbing algorithms and by branch and bound algorithms e.g. by Marco Schmidt and Klaus Schilling, in our work we have proposed a model which addresses the above mentioned situation in a way that it not only takes into account the best possible convergence mechanism (ACO), but also uses its inputs to create a neural interface and hence the technique of **NaACO** (Neural Augmented ACO).

- **Novel Introduction of Neu (Tau):** The use of intelligent form of up-gradation mechanism (τ) to formulate **Nueτ** (Neural tau)**,** has been one of the various novelties of this research. This function is different than conventional tau as it incorporates the multi-layered forward propagated ANN to come up with intelligent solutions.

- **Novel Composition of i-ACO:** An overall system of intelligent form of ACO known as **i-ACO** (Intelligent ACO) is thus introduced. Moreover we have first converged through the use of ψ through ACO and then we have maintained the essence of this convergence throughout our optimization process by including the key inputs in the development of the complete model.

- **Novel Creation of a Model for Redundancy Evaluation:** The key findings are that even though static scheduling has been worked upon since so many years, but if we talk about a system in which the push effect is getting dominant as compared to the pull effect, we have to take quick decisions in order to repeat the process lines or the serving lines. This effect can be nullified if we have a proper intelligent system in place. This system can advise us on the acceptability to repeat any process or assembly line within the scheduling domain.

- **The true nature of this finding is that through the usage of the proposed model the viability of repetitions at process and sub process levels can be predicted:** This viability is based on numerous factors. The factor of efficiency and effectiveness both are important and pertinent. If we look at our second case study, we see that a parallel assembly line was built using this approach in which our model hinted that the parameters which we were trying to handle were converging to take this decision. We have to be rational in our approach while commenting on the layouts and giving decision about the inclusion of another repeated setup for a process. This of course depends upon the financial limitations, the organizational culture and the amount of gain which we are expected. As shown by the snap shots taken before and after the inclusion of a parallel process line we can comment that this strategy had other various positive effects. For example the local housekeeping of the workshop drastically improved and we saw that the overall layout of the workshop became orderly. The concept of lean and waste reduction is an offshoot of the effects of this nature.

8.2 Areas of Future Research

Although NaACO presents a host of other futuristic applications and improvements to cater for changing environments, the following shows the vivid list of futuristic assignments which can be immediately pondered upon by interested researchers. The areas of research are also indicated in the relevant chapter discussions for interested readers.

- **Weight Updating Mechanism:** Although NaACO is modelled keeping in view the aspects of run time adjustments for its initiation and subsequent usage, the issue of giving weights to different inputs in response to the convergence rate is an area which can be researched especially in the presence of evolving variables. A weight assignment matrix and methodology can be constructed which will enable the model to recognize the importance of each variable.
- **Combination of NaACO with DSM (Design Structure Matrix):** The DSM or Design Structure Matrix is one area of intricate scheduling at process level in scheduling. It may be quiet interesting to dove tail this technique of scheduling with NaACO to see the combined convergence of the overall algorithm in scheduling related projects/processes.

- **Unsupervised NaACO:** The area of unsupervised neural learning amalgamated with ACO to formulate an unsupervised NaACO is another advancement for future research. In this regards SOM (Self Organizing Maps) topology can be used in applied case studies where even the outputs are unknown to generate a position vector field directing towards a particular output.

- **Transferability Function:** Any agent is regarded as intelligent once it is able to transfer the actions used in the previous encounters to the next encounters and situations. We can also regard this transfer as a pre cursor to reusability. The NaACO approach to scheduling can be further sufficed by adding a *mathematical formulation of reusability function* in it.
- **Recoverability Function:** Every intelligent system should be able to identify the aspects which made it to come to a conclusion hence the causes of its failure and the causes for its success. This ability enables the users to clearly identify the agents which are causing disruption and enabling the system to diverge. This is also true for over convergence. This is known as recoverability of a system. Hence NaACO can be reinforced with the development of a *recoverability mechanism.*

APPENDIX A

Data sets for 100 problems

The selected data sets of 100 problems used in the problem formulation phase (chapter 5) are presented in this appendix. There are 12 jobs which are to be scheduled on three machines with times as A, B and E for each machine.

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	90	188	190	80	185	186	77	190	178
2	87	192	185	87	193	188	87	188	191
3	89	190	188	88	192	190	88	189	188
4	90	185	184	87	188	185	76	188	190
5	86	183	187	82	190	189	70	190	191
6	70	187	190	92	188	195	80	188	190
7	87	193	187	87	190	185	98	191	188
8	88	190	188	91	185	183	70	185	183
9	76	188	184	88	188	190	88	188	190
10	89	189	190	70	190	184	89	185	191
11	76	190	183	60	192	190	90	188	190
12	85	192	189	78	187	188	78	190	191

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	9	263	9	0	205	5	4	164	4
2	6	76	6	1	71	4	3	691	5
3	5	471	4	3	24	8	1	470	8
4	3	135	2	1	258	8	7	603	5
5	1	742	5	0	632	7	9	742	4
6	7	78	2	5	237	7	5	264	5
7	4	355	9	6	188	9	0	434	3
8	7	218	2	5	384	4	7	64	6
9	5	698	1	8	203	5	4	507	5
10	8	600	4	0	272	4	4	328	7
11	0	218	5	1	35	7	4	768	8
12	2	538	9	6	385	2	2	91	4

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	3	596	6	6	587	8	1	767	2
2	2	70	9	7	221	1	1	433	9
3	1	507	6	1	285	5	0	395	8
4	5	570	4	8	346	7	7	778	7
5	2	12	7	2	755	8	5	174	4
6	5	344	1	9	97	3	5	303	1
7	3	321	7	0	516	9	2	316	2
8	8	220	7	0	278	8	4	225	2
9	4	788	5	7	83	7	7	402	5
10	1	642	2	3	148	9	7	110	4
11	6	556	3	4	62	8	3	413	1
12	7	334	3	1	346	4	9	772	3

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	2	336	2	4	143	8	3	560	9
2	5	411	3	7	139	4	6	664	7
3	0	3	6	1	773	7	1	69	5
4	8	532	3	4	555	4	9	654	8
5	4	781	3	8	708	4	3	650	4
6	5	74	1	5	101	3	8	161	2
7	3	624	1	7	120	3	1	83	9
8	1	35	9	2	215	1	4	499	5
9	7	772	7	0	25	8	0	60	8
10	2	799	4	6	259	8	7	498	6
11	2	550	5	5	721	3	8	369	4
12	3	550	5	6	191	4	2	754	9

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	3	52	7	8	694	1	1	593	8
2	3	10	7	1	686	3	3	568	5
3	2	764	4	7	332	9	4	147	8
4	7	167	2	4	476	1	9	46	8
5	8	714	6	5	151	8	1	196	1
6	1	356	2	0	755	1	8	158	6
7	9	135	3	0	206	7	6	780	5
8	6	230	8	8	14	3	2	141	8
9	8	600	7	9	257	9	0	524	4
10	7	683	5	0	480	1	9	727	9
11	3	204	6	6	303	6	4	217	2
12	5	277	3	9	513	5	6	205	7

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	6	261	3	6	537	4	7	575	9
2	7	210	3	1	185	7	8	204	5
3	4	692	2	6	554	2	3	698	9
4	9	207	3	3	262	9	8	625	4
5	8	126	5	0	38	3	7	371	7
6	1	128	9	6	755	1	3	100	7
7	6	193	1	5	218	3	5	159	2
8	2	141	7	5	539	6	3	195	2
9	8	419	1	8	798	2	5	796	6
10	7	769	6	9	303	8	4	12	9
11	3	315	7	2	292	2	5	262	7
12	0	238	8	1	366	4	7	569	6

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	2	722	7	4	29	7	7	81	3
2	1	637	1	9	145	4	5	346	5
3	5	376	6	5	564	3	0	290	5
4	9	653	1	5	555	3	4	196	3
5	0	292	5	8	590	8	5	211	5
6	0	717	7	7	413	8	6	117	3
7	4	766	7	2	513	5	1	700	2
8	6	194	2	0	252	5	7	794	1
9	8	538	1	4	665	6	6	331	3
10	5	372	5	0	490	6	6	618	8
11	1	24	7	1	375	7	1	12	6
12	3	759	8	4	590	3	2	617	7

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	6	47	3	3	133	5	0	618	9
2	8	740	1	1	66	8	6	270	2
3	6	35	9	7	710	5	1	514	8
4	6	778	8	7	197	9	2	130	7
5	2	12	2	9	424	3	5	785	1
6	4	270	7	3	424	8	0	261	3
7	9	172	7	5	132	1	7	63	5
8	6	388	6	4	636	4	4	723	6
9	1	525	1	2	797	4	2	693	7
10	8	107	9	3	43	6	3	485	3
11	0	727	9	7	67	9	0	552	3
12	7	321	3	2	663	9	0	774	4

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	3	335	7	6	569	5	1	790	7
2	6	330	5	3	114	4	8	367	3
3	6	495	1	3	286	4	2	587	6
4	3	701	8	3	501	1	9	224	4
5	4	130	7	9	617	3	3	173	4
6	3	55	4	6	340	3	3	495	9
7	7	194	4	2	305	7	5	768	9
8	9	383	8	7	159	9	7	577	6
9	5	146	2	8	301	9	5	620	5
10	1	619	9	0	710	6	6	434	2
11	2	68	8	5	331	1	5	511	4
12	8	268	6	9	71	6	7	475	9

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	0	466	3	5	570	8	1	208	5
2	3	0	7	7	797	8	4	624	9
3	2	504	5	2	179	6	0	283	6
4	1	424	9	9	594	2	7	533	3
5	5	116	7	7	216	5	5	250	2
6	3	234	1	3	305	1	0	450	4
7	6	49	1	7	127	7	2	346	8
8	2	442	5	3	739	7	0	316	1
9	9	299	3	0	15	4	9	381	6
10	3	771	1	2	401	5	9	213	6
11	2	179	8	1	749	8	2	792	1
12	8	636	1	6	342	4	0	781	8

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	3	783	5	8	487	1	5	737	8
2	1	258	5	8	709	5	7	302	1
3	7	703	4	0	458	1	4	454	2
4	6	709	4	6	318	1	3	610	7
5	9	414	4	8	761	9	7	451	1
6	1	481	3	4	114	2	5	746	5
7	1	477	1	8	159	3	5	558	2
8	4	251	5	7	465	6	4	179	9
9	5	597	5	9	377	6	3	698	4
10	7	505	2	8	498	9	7	145	8
11	7	493	2	2	716	7	5	542	3
12	7	146	7	6	269	8	6	288	4

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	3	237	6	7	232	4	8	591	4
2	4	182	3	8	166	1	2	584	3
3	8	268	4	9	214	8	0	676	2
4	4	600	5	5	64	7	6	535	2
5	4	784	4	3	283	9	1	302	6
6	5	645	3	9	275	5	8	520	9
7	6	771	2	7	102	6	3	653	8
8	3	13	1	5	759	6	0	646	1
9	5	775	1	1	526	9	8	42	9
10	8	578	5	4	761	9	9	476	9
11	3	656	9	3	799	9	9	262	4
12	2	710	4	8	402	9	2	287	6

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	0	36	8	2	765	8	3	369	6
2	7	698	3	3	701	8	2	411	2
3	1	587	1	6	202	4	1	513	9
4	4	442	9	7	126	3	0	674	9
5	6	463	3	4	27	7	6	140	7
6	9	793	8	4	452	3	5	243	7
7	9	369	3	4	538	1	4	578	1
8	7	295	7	7	548	2	4	395	6
9	5	310	2	5	262	4	1	627	8
10	2	34	4	6	237	3	4	33	9
11	3	103	9	1	712	6	6	60	9
12	3	117	2	0	36	2	6	793	1

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	7	309	8	0	238	6	8	161	7
2	1	19	7	9	378	8	2	231	7
3	2	402	5	3	292	4	9	60	9
4	9	261	1	2	698	8	0	1	4
5	5	614	7	3	386	3	7	47	3
6	8	516	7	1	155	7	6	253	5
7	1	478	6	4	495	8	7	354	4
8	8	341	8	0	80	1	7	645	5
9	7	451	1	9	64	5	6	138	8
10	7	169	4	9	645	6	2	514	4
11	2	222	5	0	652	6	2	570	2
12	3	394	3	1	528	9	5	402	7

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	9	696	9	2	185	7	1	70	1
2	3	377	7	3	359	6	4	179	3
3	9	174	2	1	683	8	2	192	8
4	3	612	7	8	718	4	2	279	3
5	6	361	5	0	540	7	5	195	9
6	3	278	6	4	448	5	7	668	6
7	1	781	3	9	349	1	6	332	7
8	7	458	5	5	523	3	9	45	8
9	8	21	5	6	121	3	6	541	6
10	6	696	5	6	112	1	7	100	7
11	9	729	9	8	729	1	0	30	1
12	2	409	9	8	661	4	1	764	8

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	9	410	7	9	724	7	9	106	4
2	6	194	8	9	725	5	5	681	2
3	8	639	3	5	26	5	5	55	6
4	8	264	2	4	564	7	1	589	3
5	5	657	6	0	179	4	4	367	4
6	4	25	5	9	792	4	1	37	4
7	7	421	1	2	734	5	3	165	7
8	1	30	5	4	714	2	1	319	6
9	7	136	7	2	303	9	8	398	9
10	0	662	1	2	503	5	4	89	3
11	6	699	7	3	749	3	5	580	6
12	6	97	5	3	225	4	5	762	9

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	0	184	2	2	375	6	2	412	7
2	3	544	4	5	497	2	3	691	1
3	2	376	8	9	336	7	9	242	3
4	2	100	9	7	557	9	7	780	2
5	0	614	5	7	432	1	6	541	7
6	0	759	6	2	741	3	9	279	2
7	0	212	8	9	160	8	0	153	6
8	0	142	3	4	420	8	5	416	4
9	5	435	1	9	637	8	4	524	7
10	0	55	7	4	146	7	5	659	7
11	2	647	7	3	488	1	5	631	9
12	2	66	5	3	798	2	7	728	1

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	6	152	7	8	201	6	9	631	1
2	8	389	1	9	173	7	6	141	6
3	6	589	8	6	451	9	6	573	6
4	2	8	5	8	189	7	6	339	7
5	2	646	8	8	606	1	2	0	5
6	9	398	8	2	408	7	2	94	7
7	1	293	2	9	396	7	8	338	2
8	4	308	3	2	197	5	9	774	3
9	5	115	3	4	448	2	3	508	7
10	1	286	5	0	683	8	8	531	6
11	5	350	6	0	365	7	0	540	6
12	5	374	2	2	513	8	1	627	1

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	4	282	4	8	470	8	9	274	1
2	6	412	5	2	401	5	6	72	8
3	9	87	4	2	659	5	9	198	5
4	9	414	1	1	89	6	6	14	9
5	7	333	6	7	138	3	9	529	4
6	1	224	4	0	123	9	0	640	2
7	1	157	6	8	717	6	9	290	9
8	3	287	5	3	32	5	1	141	9
9	1	658	1	4	149	1	6	736	4
10	3	415	4	1	768	4	9	57	8
11	3	279	9	4	152	8	7	528	7
12	7	436	6	3	225	2	1	159	9

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	1	283	6	5	5	3	7	127	6
2	2	211	7	4	154	4	9	573	4
3	3	282	9	8	530	3	1	718	4
4	5	54	3	5	480	3	2	188	4
5	4	533	6	9	350	4	8	211	1
6	3	462	9	3	727	2	9	788	1
7	9	731	6	9	586	7	6	467	3
8	9	100	6	2	788	8	9	752	9
9	3	410	1	1	587	7	4	77	2
10	6	182	2	9	726	9	3	711	4
11	4	513	2	6	251	2	6	733	7
12	4	129	8	5	601	7	1	47	3

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	0	762	1	5	367	5	7	591	6
2	5	557	4	0	551	2	3	318	9
3	3	102	1	4	286	6	0	271	2
4	7	223	7	1	569	9	7	164	2
5	0	138	3	0	209	7	2	176	8
6	5	310	9	3	567	1	8	216	3
7	5	189	7	1	208	6	5	16	1
8	6	516	5	5	291	2	7	400	6
9	0	197	4	2	380	5	5	448	2
10	3	721	8	8	652	3	7	656	1
11	8	596	6	2	765	4	6	730	9
12	5	177	2	0	387	2	6	360	8

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	5	454	1	3	339	1	5	240	5
2	1	140	6	1	720	7	9	82	5
3	6	568	8	6	321	7	1	43	9
4	7	374	1	1	438	7	9	445	2
5	3	639	6	0	682	2	5	537	5
6	3	744	5	7	509	1	8	320	8
7	5	279	7	7	301	9	7	148	6
8	9	69	3	0	261	2	6	278	5
9	2	468	3	2	522	8	4	795	5
10	2	719	9	8	61	3	5	474	2
11	0	570	8	5	630	8	9	293	1
12	8	355	3	5	114	3	2	619	9

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	4	790	9	5	742	2	1	253	2
2	1	465	2	8	499	1	3	633	3
3	7	191	5	1	224	7	7	421	6
4	3	314	9	0	413	2	0	227	4
5	9	198	5	1	699	1	4	650	4
6	5	536	5	7	431	8	2	603	1
7	5	500	7	1	419	8	7	424	3
8	1	425	6	4	50	9	6	675	7
9	9	148	3	4	520	6	8	535	9
10	2	31	3	6	172	5	1	665	9
11	7	177	4	8	579	1	9	495	6
12	3	710	1	7	22	8	9	697	1

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	8	790	5	1	680	2	7	380	4
2	0	78	9	4	301	3	0	622	1
3	3	714	8	1	171	5	9	748	3
4	2	453	7	4	710	7	8	561	3
5	9	47	3	6	792	4	1	297	6
6	0	784	7	0	201	4	4	317	4
7	4	166	3	1	761	8	0	324	1
8	5	797	9	3	675	3	5	451	3
9	7	722	6	9	701	8	6	177	4
10	9	240	2	0	465	4	4	625	6
11	0	756	1	3	198	5	6	552	5
12	1	752	4	2	272	2	9	510	4

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	0	244	6	2	579	6	9	682	5
2	0	228	5	0	541	3	1	675	3
3	9	734	3	0	444	4	2	104	4
4	6	592	2	5	116	4	8	741	1
5	4	635	4	1	305	5	0	212	8
6	8	6	7	5	473	5	2	7	1
7	8	342	5	1	310	5	0	547	1
8	9	341	4	8	124	3	5	306	1
9	0	269	8	8	59	2	2	643	9
10	5	661	4	2	477	2	1	739	7
11	7	711	5	5	303	1	7	476	9
12	4	437	3	6	331	9	5	738	6

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	4	204	9	5	718	9	4	451	3
2	1	612	4	5	343	1	9	690	6
3	1	184	3	3	183	9	4	296	1
4	7	231	2	4	66	8	5	62	5
5	8	352	3	8	352	7	0	762	8
6	5	756	5	6	435	2	6	213	9
7	4	593	6	0	716	6	0	561	7
8	3	687	7	1	233	2	9	567	2
9	2	536	1	4	353	7	3	345	7
10	6	544	8	9	733	3	1	774	4
11	8	409	7	6	724	5	7	349	7
12	7	700	3	7	105	1	9	182	4

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	8	617	2	2	651	1	7	538	1
2	9	617	2	1	489	2	3	628	2
3	3	240	2	7	503	1	7	482	8
4	8	457	7	3	627	3	1	102	1
5	4	697	5	0	309	9	6	248	2
6	2	449	9	5	245	8	0	339	6
7	6	730	8	9	442	6	0	231	1
8	2	29	8	9	184	3	8	416	3
9	2	658	5	0	510	6	7	26	6
10	0	741	2	5	26	8	6	649	8
11	3	331	1	9	148	4	3	570	8
12	4	795	6	8	121	8	4	585	5

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	50	188	190	47	185	186	50	190	178
2	46	192	185	46	193	188	48	188	191
3	45	190	188	50	192	190	50	189	188
4	44	185	184	49	188	185	51	188	190
5	51	183	187	55	190	189	47	190	191
6	47	187	190	44	188	195	46	188	190
7	49	193	187	50	190	185	45	191	188
8	51	190	188	51	185	183	51	185	183
9	50	188	184	46	188	190	48	188	190
10	47	189	190	48	190	184	50	185	191
11	48	190	183	49	192	190	51	188	190
12	52	192	189	50	187	188	50	185	191

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	1	535	2	4	572	4	3	294	3
2	0	620	6	1	560	1	5	25	7
3	5	758	8	0	582	2	6	500	1
4	9	321	9	3	110	9	3	422	9
5	6	649	1	1	95	9	3	364	5
6	0	159	7	0	382	4	8	565	8
7	9	300	7	5	547	4	4	30	1
8	2	221	1	4	527	2	0	176	4
9	0	771	4	1	513	6	7	667	9
10	3	527	2	2	722	6	5	359	8
11	4	391	9	5	548	8	6	341	1
12	2	550	2	8	62	1	0	366	6

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	0	735	1	2	503	7	1	521	3
2	0	296	5	4	48	1	7	774	5
3	3	704	7	8	86	9	3	160	2
4	5	157	5	1	299	9	3	321	5
5	1	322	2	5	26	7	1	628	9
6	5	199	1	3	501	1	1	487	1
7	2	400	8	7	254	9	1	319	6
8	9	652	7	6	215	3	9	548	8
9	3	283	2	2	18	3	5	435	5
10	7	286	5	9	705	4	3	603	2
11	5	234	5	7	109	5	0	760	8
12	6	365	3	7	249	6	0	258	7

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	6	271	5	6	479	5	3	216	1
2	8	568	7	3	722	2	0	464	2
3	4	249	8	3	459	9	7	167	4
4	1	639	4	8	196	5	8	62	2
5	8	121	2	5	688	1	2	716	9
6	3	474	3	9	59	4	7	89	8
7	3	764	4	5	351	3	4	521	6
8	7	194	2	3	607	7	8	720	7
9	2	751	6	8	196	9	7	185	2
10	4	91	1	2	302	8	4	759	3
11	2	787	2	2	317	2	0	676	1
12	8	506	7	1	421	1	4	352	8

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	7	248	4	5	516	2	9	11	6
2	7	236	1	4	412	3	2	235	5
3	3	646	5	5	178	1	1	787	6
4	4	247	7	0	466	2	6	609	3
5	5	300	5	5	598	4	0	464	9
6	8	320	4	2	365	5	3	484	7
7	5	673	9	9	723	6	6	214	6
8	2	327	9	9	226	8	1	406	4
9	2	579	6	9	534	1	1	634	7
10	3	527	5	8	712	4	2	685	2
11	5	325	6	4	225	1	3	74	1
12	3	760	1	8	294	3	5	433	3

	Machine 1			Machine 2			Machine 3		
Job	A_i	B_i	E_i	A_i	B_i	E_i	A_i	B_i	E_i
1	7	521	6	6	655	1	5	659	5
2	9	646	8	0	175	6	3	575	2
3	3	333	6	6	147	9	4	716	3
4	6	637	4	9	189	4	5	178	1
5	7	576	2	0	740	2	7	284	9
6	0	554	5	9	206	4	4	442	3
7	7	584	9	0	204	7	3	284	6
8	4	153	7	2	571	4	0	163	3
9	2	618	4	5	531	4	4	463	8
10	6	55	5	8	618	7	5	503	3
11	9	198	5	0	607	4	7	111	8
12	9	760	7	0	246	7	5	48	2

APPENDIX B

This appendix gives the codes used during the formulation and validation of NaACO and also for the applied case study.

Code for PC Hu problem sets (used during formulation of NaACO)

```
learall
closeall
clc

importfile('PC Hu_Min_Makespan_LV.xls');
Problem_No=0;
tic
Span_Arr=[];
tmp_r = 16;
for r2 = 5:15:140
    tmp_c = 12;
for r = 4:11:103
        W1 = 1;%input('W1 = ');
        W2 = 1;%input('W2 = ');
        W3 = 1;%input('W3 = ');
        Problem_No=Problem_No+1
table = data(r2:tmp_r,r:tmp_c);
        machine1 = table(:,1:3);
        machine2 = table(:,4:6);
        machine3 = table(:,7:9);

        TT1 = machine1(:,1) + (machine1(:,2)./(machine1(:,3)*W1));
        eta1 = 1./TT1;

        TT2 = machine2(:,1) + (machine2(:,2)./(machine2(:,3)*W2));
        eta2 = 1./TT2;
        TT3 = machine3(:,1) + (machine3(:,2)./(machine3(:,3)*W3));
        eta3 = 1./TT3;

        P1 = eta1./(eta2+eta3);
        P2 = eta2./(eta1+eta3);
        P3 = eta3./(eta1+eta2);

        P = [P1 P2 P3];

        M_num=[];
for j = 1:12
            M_num(end+1) = find(max(P(j,:)) == P(j,:));
end

        M_num;
display('Jobs Assigned');
        Station_1 = find(M_num==1)
        Station_2 = find(M_num==2)
        Station_3 = find(M_num==3)
        FT1= sum(TT1(Station_1));
        FT2= sum(TT2(Station_2));
        FT3= sum(TT3(Station_3));
        SFT = FT1+FT2+FT3;
display('Workers Assigned');
        M_1 = round((FT1/SFT)*10)
        M_2 = round((FT2/SFT)*10)
        M_3 = 10 - (M_1 + M_2)

        ext_jobs_m1 = machine1(Machine_1,:);
        [r,c] = size(ext_jobs_m1);
flowtime = 0;
oldflowtime=0;
        m1_flowtime=[];
for p = 1:r
flowtime = oldflowtime + ext_jobs_m1(p,1)+
(ext_jobs_m1(p,2)/(ext_jobs_m1(p,3)*M_1));
```

```
oldflowtime = flowtime;
            m1_flowtime(end+1) = flowtime;
end
        total_flowtime_machine_1 = flowtime;
flowtime = 0;
        ext_jobs_m2 = machine2(Machine_2,:);
        [r,c] = size(ext_jobs_m2);
oldflowtime=0;
        m2_flowtime=[];
for p = 1:r
flowtime = oldflowtime + ext_jobs_m2(p,1)+
(ext_jobs_m2(p,2)/(ext_jobs_m2(p,3)*M_2));
oldflowtime = flowtime;
            m2_flowtime(end+1) = flowtime;
end


        total_flowtime_machine_2 = flowtime;
flowtime = 0;


        ext_jobs_m3 = machine3(Machine_3,:);
        [r,c] = size(ext_jobs_m3);
oldflowtime=0;
        m3_flowtime=[];
for p = 1:r
flowtime = oldflowtime + ext_jobs_m3(p,1)+
(ext_jobs_m3(p,2)/(ext_jobs_m3(p,3)*M_3));
oldflowtime = flowtime;
            m3_flowtime(end+1) = flowtime;
end
        total_flowtime_machine_3 = flowtime;
        Make_span =
max(max(total_flowtime_machine_1,total_flowtime_machine_2),total_flowtime_mac
hine_3)
        Span_Arr(end+1) = Make_span;
        tmp_c = tmp_c+11;


end
```

```
    tmp_r = tmp_r+15;
end
tmp_arr = [1:100]';
[tmp_arr Span_Arr'];
toc
```

Code for validation of FT06 (ACO Domain)

```
clearall
closeall
jobs = 6;
no_machines = 6;
machines = [3 1 2 4 6 5;
            2 3 5 6 1 4;
            3 4 6 1 2 5;
            2 1 3 4 5 6;
            3 2 5 6 1 4;        % for 6 jobs
            2 4 6 1 5 3]
time = [1 3 6 7 3 6;
        8 5 10 10 10 4;
        5 4 8 9 1 7;
        5 5 5 3 8 9;
        9 3 5 4 3 1;
        3 3 9 10 4 1]

eta =1./time;
prob = zeros(jobs,no_machines);
avg_time = sum(sum(time))/(jobs*no_machines);
tau = no_machines/avg_time;
for i = 1:jobs
for j = 1:no_machines
        prob(i,j) = eta(i,j)/(sum(eta(i,:))-eta(i,j));
end
end
sorted_time = zeros(jobs,no_machines);
sorted_machines = zeros(jobs,no_machines);
```

```
sorted_prob_mat = zeros(jobs,no_machines);
for i = 1:jobs
    [sorted_prob,sorted_locations] = sort(prob(i,:),'descend');
    sorted_machines(i,:) = machines(i,sorted_locations);
    sorted_time(i,:) = time(i,sorted_locations);
    sorted_prob_mat(i,:) = sorted_prob;
end
sorted_machines
sorted_time
sorted_prob_mat
makespan_g =100;
final_machines = zeros(jobs,makespan_g);
final_machines_con = zeros(jobs,makespan_g,no_machines);
for i = 1:no_machines
    machine_col = sorted_machines(:,i);
    machine_col_n = machine_col;
    prob_col = sorted_prob_mat(:,i);
    time_col = sorted_time(:,i);
    u_machines = unique(machine_col);
    a=1; %end location
    loc_arr = [];
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==1)  % no repetition
            t = time_col(locs);
for p = 1:t
                final_machines(locs,a+p-1) = machine_col(locs);
end
            loc_arr(end+1) = locs;

% machine run
end


end
    machine_col_n(loc_arr)=[];
    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
```

```
if (length(locs)==2)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc] = min(prob_col(locs));
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
end
% machine run

            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end


end


    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==3)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
            tu1=0;
            tu2=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(2);
            t = time_col(locs(loc));
for p = 1:t
```

```
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
                tu1= tu1+1;
end
            [mv,loc] = min(prob_col(locs));
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1) =
machine_col(locs(loc));
                tu2= tu2+1;
end
% machine run
            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end


end


final_machines_con(:,:,i) = final_machines;
end


final_row=[];
final_mat =zeros(jobs,makespan_g);
makespan=[];
for i = 1 : jobs
for j = 1: no_machines
row = final_machines_con(i,:,j);
        [I,J] = find(row~=0);
row = row(:,1:max(J));
        final_row = [final_row row];


end
makespan(end+1) = length(final_row);
    final_row = [];
end


max(makespan)
min(makespan)
```

```
%seq_machines = [3 6 1 2 5 4;
%             4 3 2 6 1 5;
%             2 4 3 5 6 1;
%             4 3 1 2 5 6;
%             4 2 1 6 5 3;
%             3 2 4 5 6 1];
```

Code for validation of FT10 (ACO Domain)

```
clearall
closeall
jobs = 10;
no_machines = 10;
machines = [1 2 3 4 5 6 7 8 9 10;
            1 3 5 10 4 2 7 6 8 9;
            2 1 4 3 9 6 8 7 10 5;
            2 3 1 5 7 9 8 4 10 6;
            3 1 2 6 4 5 9 8 10 7;
            3 2 6 4 9 10 1 7 5 8;
            2 1 4 3 7 6 10 9 8 5;
            3 1 2 6 5 7 9 10 8 4;
            1 2 4 6 3 10 7 8 5 9;
            2 1 3 7 9 10 6 4 5 8]
time = [29 78 9 36 49 11 62 56 44 21;
        43 28 90 69 75 46 46 72 30 11;
        85 91 74 39 33 10 89 12 90 45;
        71 81 95 98 99 43 9 85 52 22;
        6 22 14 26 69 61 53 49 21 72;
        47 2 84 95 6 52 65 25 48 72;
        37 46 13 61 55 21 32 30 89 32;
        86 46 31 79 32 74 88 36 19 48;
        76 69 85 76 26 51 40 89 74 11;
        13 85 61 52 90 47 7 45 64 76]

eta =1./time;
```

```
prob = zeros(jobs,no_machines);
avg_time = sum(sum(time))/(jobs*no_machines);
tau = no_machines/avg_time;
for i = 1:jobs
for j = 1:no_machines
        prob(i,j) = eta(i,j)/(sum(eta(i,:))-eta(i,j));
end
end
sorted_time = zeros(jobs,no_machines);
sorted_machines = zeros(jobs,no_machines);
sorted_prob_mat = zeros(jobs,no_machines);
for i = 1:jobs
    [sorted_prob,sorted_locations] = sort(prob(i,:),'descend');
    sorted_machines(i,:) = machines(i,sorted_locations);
    sorted_time(i,:) = time(i,sorted_locations);
    sorted_prob_mat(i,:) = sorted_prob;
end
sorted_machines
sorted_time
sorted_prob_mat
makespan_g =1000;
final_machines = zeros(jobs,makespan_g);
final_machines_con = zeros(jobs,makespan_g,no_machines);
for i = 1:no_machines
    machine_col = sorted_machines(:,i);
    machine_col_n = machine_col;
    prob_col = sorted_prob_mat(:,i);
    time_col = sorted_time(:,i);
    u_machines = unique(machine_col);
    a=1; %end location
    loc_arr = [];
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==1)  % no repetition
            t = time_col(locs);
for p = 1:t
                final_machines(locs,a+p-1) = machine_col(locs);
end
```

```
            loc_arr(end+1) = locs;

% machine run
end


end
    machine_col_n(loc_arr)=[];
    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==2)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc] = min(prob_col(locs));
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
end
% machine run

            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end


end


    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==3)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
```

```
            tu1=0;
            tu2=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(2);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
                tu1= tu1+1;
end
            [mv,loc] = min(prob_col(locs));
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1) =
machine_col(locs(loc));
                tu2= tu2+1;
end
% machine run
            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end


end


final_machines_con(:,:,i) = final_machines;
end


final_row=[];
final_mat =zeros(jobs,makespan_g);
makespan=[];
for i = 1 : jobs
for j = 1: no_machines
row = final_machines_con(i,:,j);
        [I,J] = find(row~=0);
row = row(:,1:max(J));
```

```
        final_row = [final_row row];

end
makespan(end+1) = length(final_row);
    final_row = [];
end

max(makespan)
min(makespan)
(max(makespan)+min(makespan))/2

%seq_machines = [3 6 1 2 5 4;
%             4 3 2 6 1 5;
%             2 4 3 5 6 1;
%             4 3 1 2 5 6;
%             4 2 1 6 5 3;
%             3 2 4 5 6 1];
```

Code for validation of FT10 (ANN Domain)

```
clc
clearall
closeall
eg1=[3;6;10;1;4;9;5;8;7;2;395]
eg2=[9;3;8;1;2;7;10;6;4;5;464]
eg3=[6;7;9;3;5;4;2;8;10;1;568]
eg4=[8;6;9;10;2;3;4;1;5;7;655]
eg5=[3;2;10;1;6;8;9;5;4;7;393]
eg6=[2;9;7;3;5;10;1;8;6;4;496]
eg7=[4;6;9;10;5;2;1;7;3;8;416]
eg8=[8;2;5;10;1;4;7;6;3;9;539]
eg9=[9;3;7;10;2;5;1;6;4;8;597]
eg10=[6;2;4;10;7;3;5;8;1;9;540]
eg11=[10;3;6;1;4;9;5;8;7;2;250]
eg12=[8;3;10;1;2;7;10;6;4;5;280]
eg13=[9;7;10;3;5;4;2;8;6;1;400]
```

```
eg14=[9;6;8;10;2;3;4;1;5;7;600]
eg15=[10;3;2;1;6;8;9;5;4;7;250]

in=[eg1 eg2 eg3 eg4 eg5 eg6 eg7 eg8 eg9 eg10 eg11 eg12 eg13 eg14 eg15];

Target=[1 1 1 1 1 1 1 1 1 1 0 0 0 0 0]

net = newff(minmax(in),[10 10 10 1]);
net.trainParam.epochs = 500;
net = train(net,in,Target);
savenetnet
```

Code for validation of FT20 (ACO Domain)

```
clearall
closeall
jobs = 20;
no_machines = 5;
machines = [1 2 3 4 5;
            1 2 4 3 5;
            2 1 3 5 4;
            2 1 5 3 4;
            3 2 1 4 5;
            3 2 5 1 4;
            2 1 3 4 5;
            3 2 1 4 5;
            1 4 3 2 5;
            2 3 1 4 5;
            2 4 1 5 3;
            3 1 2 4 5;
            1 3 2 4 5;
            3 1 2 4 5;
            1 2 5 3 4;
            2 1 4 5 3;
            1 3 2 4 5;
            1 2 5 3 4;
            2 3 1 4 5;
            1 2 3 4 5]
time = [29 9 49 62 44;
        43 75 46 69 72;
        39 91 90 45 12;
        71 81 85 22 9;
        26 22 14 21 72;
        47 52 84 6 48;
        61 46 32 32 30;
        32 46 31 19 36;
        76 40 85 76 26;
        64 85 61 47 90;
        11 78 21 36 56;
        11 28 90 46 30;
        85 10 74 89 33;
        99 52 95 98 43;
        6 61 49 53 69;
        95 2 25 72 65;
        37 21 13 89 55;
        86 74 48 79 88;
        11 69 51 89 74;
        13 7  76 52 45]

eta =1./time;
prob = zeros(jobs,no_machines);
avg_time = sum(sum(time))/(jobs*no_machines);
tau = no_machines/avg_time;
for i = 1:jobs
for j = 1:no_machines
        prob(i,j) = eta(i,j)/(sum(eta(i,:))-eta(i,j));
end
end
sorted_time = zeros(jobs,no_machines);
sorted_machines = zeros(jobs,no_machines);
sorted_prob_mat = zeros(jobs,no_machines);
for i = 1:jobs
    [sorted_prob,sorted_locations] = sort(prob(i,:),'descend');
    sorted_machines(i,:) = machines(i,sorted_locations);
    sorted_time(i,:) = time(i,sorted_locations);
    sorted_prob_mat(i,:) = sorted_prob;
end
sorted_machines
sorted_time
sorted_prob_mat
makespan_g =1500;
final_machines = zeros(jobs,makespan_g);
final_machines_con = zeros(jobs,makespan_g,no_machines);
for i = 1:no_machines
    machine_col = sorted_machines(:,i);
    machine_col_n = machine_col;
    prob_col = sorted_prob_mat(:,i);
    time_col = sorted_time(:,i);
    u_machines = unique(machine_col);
    a=1; %end location
    loc_arr = [];
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==1)  % no repetition
            t = time_col(locs);
for p = 1:t
                final_machines(locs,a+p-1) = machine_col(locs);
end
            loc_arr(end+1) = locs;

% machine run
end


end
    machine_col_n(loc_arr)=[];
    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==2)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
```

```
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc] = min(prob_col(locs));
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
end
% machine run

            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end


end


    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==3)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
            tu1=0;
            tu2=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(2);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
                tu1= tu1+1;
end
            [mv,loc] = min(prob_col(locs));
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1) =
machine_col(locs(loc));
                tu2= tu2+1;
end
% machine run
            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end
end

    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==4)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
            tu1=0;
            tu2=0;
            tu3=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(2);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
                tu1= tu1+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(3);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1) =
machine_col(locs(loc));
```

```
                tu2= tu2+1;
end
            [mv,loc] = min(prob_col(locs));
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2) =
machine_col(locs(loc));
                tu3= tu3+1;
end
% machine run
            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end
end


    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==6)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
            tu1=0;
            tu2=0;
            tu3=0;
            tu4=0;
            tu5=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(2);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
                tu1= tu1+1;
end
            [mv,loc_x] = sort(prob_col(locs));
```

```
loc = loc_x(3);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1) =
machine_col(locs(loc));
                tu2= tu2+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(4);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2) =
machine_col(locs(loc));
                tu3= tu3+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(5);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3) =
machine_col(locs(loc));
                tu4= tu4+1;
end
            [mv,loc] = min(prob_col(locs));
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4) =
machine_col(locs(loc));
                tu5= tu5+1;
end
% machine run
            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end
end


    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
```

```
if (length(locs)==7)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
            tu1=0;
            tu2=0;
            tu3=0;
            tu4=0;
            tu5=0;
            tu6=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(2);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
                tu1= tu1+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(3);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1) =
machine_col(locs(loc));
                tu2= tu2+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(4);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2) =
machine_col(locs(loc));
                tu3= tu3+1;
end
            [mv,loc_x] = sort(prob_col(locs));
```

```
loc = loc_x(5);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3) =
machine_col(locs(loc));
                tu4= tu4+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(6);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4) =
machine_col(locs(loc));
                tu5= tu5+1;
end
            [mv,loc] = min(prob_col(locs));
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4+tu5) =
machine_col(locs(loc));
                tu6= tu6+1;
end
% machine run
            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end
end


    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==8)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
            tu1=0;
            tu2=0;
            tu3=0;
            tu4=0;
            tu5=0;
            tu6=0;
            tu7=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(2);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
                tu1= tu1+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(3);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1) =
machine_col(locs(loc));
                tu2= tu2+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(4);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2) =
machine_col(locs(loc));
                tu3= tu3+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(5);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3) =
machine_col(locs(loc));
                tu4= tu4+1;
end
```

```
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(6);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4) =
machine_col(locs(loc));
                tu5= tu5+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(7);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4+tu5) =
machine_col(locs(loc));
                tu6= tu6+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(8);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4+tu5+tu6) =
machine_col(locs(loc));
                tu7= tu7+1;
end
% machine run
            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end
end


    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==9)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
            tu1=0;
            tu2=0;
            tu3=0;
            tu4=0;
            tu5=0;
            tu6=0;
            tu7=0;
            tu8=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(2);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
                tu1= tu1+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(3);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1) =
machine_col(locs(loc));
                tu2= tu2+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(4);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2) =
machine_col(locs(loc));
                tu3= tu3+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(5);
            t = time_col(locs(loc));
for p = 1:t
```

```
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3) =
machine_col(locs(loc));
                tu4= tu4+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(6);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4) =
machine_col(locs(loc));
                tu5= tu5+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(7);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4+tu5) =
machine_col(locs(loc));
                tu6= tu6+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(8);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4+tu5+tu6) =
machine_col(locs(loc));
                tu7= tu7+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(9);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-
1+tu+tu1+tu2+tu3+tu4+tu5+tu6+tu7) = machine_col(locs(loc));
                tu8= tu8+1;
end
% machine run
            machine_col_n(find(machine_col_n==u_machines(k)))=[]
```

```
end
end
    u_machines = unique(machine_col_n)
for k = 1:length(u_machines)
locs = find(machine_col==u_machines(k))
if (length(locs)==10)
            [mv,loc] = max(prob_col(locs));
            t = time_col(locs(loc));
tu=0;
            tu1=0;
            tu2=0;
            tu3=0;
            tu4=0;
            tu5=0;
            tu6=0;
            tu7=0;
            tu8=0;
            tu9=0;
for p = 1:t
                final_machines(locs(loc),a+p-1) = machine_col(locs(loc));
tu= tu+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(2);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu) = machine_col(locs(loc));
                tu1= tu1+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(3);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1) =
machine_col(locs(loc));
                tu2= tu2+1;
end
            [mv,loc_x] = sort(prob_col(locs));
```

```
loc = loc_x(4);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2) =
machine_col(locs(loc));
                tu3= tu3+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(5);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3) =
machine_col(locs(loc));
                tu4= tu4+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(6);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4) =
machine_col(locs(loc));
                tu5= tu5+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(7);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4+tu5) =
machine_col(locs(loc));
                tu6= tu6+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(8);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-1+tu+tu1+tu2+tu3+tu4+tu5+tu6) =
machine_col(locs(loc));
                tu7= tu7+1;
```

```
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(9);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-
1+tu+tu1+tu2+tu3+tu4+tu5+tu6+tu7) = machine_col(locs(loc));
                tu8= tu8+1;
end
            [mv,loc_x] = sort(prob_col(locs));
loc = loc_x(10);
            t = time_col(locs(loc));
for p = 1:t
                final_machines(locs(loc),a+p-
1+tu+tu1+tu2+tu3+tu4+tu5+tu6+tu7+tu8) = machine_col(locs(loc));
                tu9= tu9+1;
end
% machine run
            machine_col_n(find(machine_col_n==u_machines(k)))=[]
end
end

final_machines_con(:,:,i) = final_machines;
end

final_row=[];
final_mat =zeros(jobs,makespan_g);
makespan=[];
for i = 1 : jobs
for j = 1: no_machines
row = final_machines_con(i,:,j);
        [I,J] = find(row~=0);
row = row(:,1:max(J));
        final_row = [final_row row];


end
makespan(end+1) = length(final_row);
```

```
    final_row = [];
end

max(makespan)
min(makespan)
(max(makespan)+min(makespan))/2

%seq_machines = [3 6 1 2 5 4;
%             4 3 2 6 1 5;
%             2 4 3 5 6 1;
%             4 3 1 2 5 6;
%             4 2 1 6 5 3;
%             3 2 4 5 6 1];
```

Code for validation of FT20 (ANN Domain)

```
clc
clearall
closeall
eg1=[1;2;3;4;5;193]
eg2=[1;2;4;3;5;305]
eg3=[2;1;3;5;4;277]
eg4=[2;1;5;3;4;268]
eg5=[3;2;1;4;5;155]
eg6=[3;2;5;1;4;237]
eg7=[2;1;3;4;5;201]
eg8=[3;2;1;4;5;164]
eg9=[1;4;3;2;5;303]
eg10=[2;3;1;4;5;347]
eg11=[2;4;1;5;3;202]
eg12=[3;1;2;4;5;205]
eg13=[1;3;2;4;5;291]
eg14=[3;1;2;4;5;387]
eg15=[1;2;5;3;4;238]
eg16=[2;1;4;5;3;259]
eg17=[1;3;2;4;5;215]
```

```
eg18=[1;2;5;3;4;375]
eg19=[2;3;1;4;5;294]
eg20=[1;2;3;4;5;193]
eg21=[1;2;3;4;5;100]
eg22=[4;2;1;3;5;250]
eg23=[3;1;2;5;4;175]
eg24=[3;1;2;4;4;200]
eg25=[2;3;1;4;5;100]
eg26=[1;2;3;5;4;175]
eg27=[4;5;3;1;2;150]
eg28=[4;2;5;3;1;90]
eg29=[5;2;3;4;1;250]
eg30=[4;5;1;2;3;275]

in=[eg1 eg2 eg3 eg4 eg5 eg6 eg7 eg8 eg9 eg10 eg11 eg12 eg13 eg14 eg15 eg16
eg17 eg18 eg19 eg20 eg21 eg22 eg23 eg24 eg25 eg26 eg27 eg28 eg29 eg30];

Target=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0]

net = newff(minmax(in),[10 10 10 1]);
net.trainParam.epochs = 500;
net = train(net,in,Target);
savenetnet
```

Code for Case Study 1 (Magneto Scheduling)

```
closeall
clc

importfile('PC Hu_Min_Makespan_LV.xls');
Problem_No=0;
tic
Span_Arr=[];
tmp_r = 16;
for r2 = 5:15:140
    tmp_c = 12;
```

```
for r = 4:11:103
        W1 = 1;%input('W1 = ');
        W2 = 1;%input('W2 = ');
        W3 = 1;%input('W3 = ');
        Problem_No=Problem_No+1
table = data(r2:tmp_r,r:tmp_c);
        machine1 = table(:,1:3);
        machine2 = table(:,4:6);
        machine3 = table(:,7:9);

        TT1 = machine1(:,1) + (machine1(:,2)./(machine1(:,3)*W1));
        eta1 = 1./TT1;

        TT2 = machine2(:,1) + (machine2(:,2)./(machine2(:,3)*W2));
        eta2 = 1./TT2;

        TT3 = machine3(:,1) + (machine3(:,2)./(machine3(:,3)*W3));
        eta3 = 1./TT3;

        P1 = eta1./(eta2+eta3);
        P2 = eta2./(eta1+eta3);
        P3 = eta3./(eta1+eta2);

        P = [P1 P2 P3];

        M_num=[];
for j = 1:12
            M_num(end+1) = find(max(P(j,:)) == P(j,:));
end

        M_num;
display('Machine Assigned');
        BAY_1 = find(M_num==1)
        BAY_2 = find(M_num==2)
        BAY_3 = find(M_num==3)
        FT1= sum(TT1(M_1));
        FT2= sum(TT2(M_2));
        FT3= sum(TT3(M_3));
        SFT = FT1+FT2+FT3;
display('Workforce Assigned');
        B_1 = round((FT1/SFT)*10)
        B_2 = round((FT2/SFT)*10)
        B_3 = 10 - (M_1 + M_2)

        ext_jobs_m1 = machine1(M_1,:);
        [r,c] = size(ext_jobs_m1);
flowtime = 0;
oldflowtime=0;
        m1_flowtime=[];
for p = 1:r
flowtime = oldflowtime + ext_jobs_m1(p,1)+
(ext_jobs_m1(p,2)/(ext_jobs_m1(p,3)*M_1));
oldflowtime = flowtime;
            m1_flowtime(end+1) = flowtime;
end
        total_flowtime_machine_1 = flowtime;
flowtime = 0;
        ext_jobs_m2 = machine2(M_2,:);
        [r,c] = size(ext_jobs_m2);
oldflowtime=0;
        m2_flowtime=[];
for p = 1:r
flowtime = oldflowtime + ext_jobs_m2(p,1)+
(ext_jobs_m2(p,2)/(ext_jobs_m2(p,3)*M_2));
oldflowtime = flowtime;
            m2_flowtime(end+1) = flowtime;
end

        total_flowtime_machine_2 = flowtime;
flowtime = 0;

        ext_jobs_m3 = machine3(M_3,:);
        [r,c] = size(ext_jobs_m3);
oldflowtime=0;
```

```
        m3_flowtime=[];
for p = 1:r
flowtime = oldflowtime + ext_jobs_m3(p,1)+
(ext_jobs_m3(p,2)/(ext_jobs_m3(p,3)*M_3));
oldflowtime = flowtime;
            m3_flowtime(end+1) = flowtime;
end
        total_flowtime_machine_3 = flowtime;
        Make_span =
max(max(total_flowtime_machine_1,total_flowtime_machine_2),total_flowtime_mac
hine_3)
        Span_Arr(end+1) = Make_span;
        tmp_c = tmp_c+11;


end
    tmp_r = tmp_r+15;
end
tmp_arr = [1:100]';
[tmp_arr Span_Arr'];
toc
```

REFERENCES

[1] Bensmaine, A., L. Benyoucef, and M. Dahane. *Process plan generation in reconfigurable manufacturing systems using adapted NSGA-II and AMOSA.* in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*. 2011. IEEE.

[2] Eiben, A., *E., Smith, J., E.* Introduction to evolutionary computing, 1998.

[3] Krishna, A.G. and K.M. Rao, *Optimisation of operations sequence in CAPP using an ant colony algorithm.* The International Journal of Advanced Manufacturing Technology, 2006. **29**(1-2): p. 159-164.

[4] Tariq, A., *Operational Design of a Cellular Manufacturing System*. 2010, National University of Sciences & Technology, Rawalpindi.

[5] Kazem, B.I., A.I. Mahdi, and A.T. Oudah, *Motion planning for a robot arm by using genetic algorithm.* JJMIE, 2008. **2**(3).

[6] Baqai, A., et al. *Algorithmic Design Methodology for Process Plans and Architectural Configurations of Manufacturing Systems", CIRPMS09*. in *42nd CIRP Conference on Manufacturing Systems*. 2009.

[7] Beale H M, Hagan T M, Demuth B H. Neural Network Toolbox™ User's Guide R2013b. MATLAB. 2013.

[8] Kordoghli, B., S. Saadallah, and M. Jmali, *Scheduling Optimization in a Cloth Manufacturing Factory Using Genetic Algorithm with Fuzzy Logic for Multi-Objective Decisions.* Journal of Textile and Apparel, Technology and Management, 2010. **6**(3).

[9] Kumar, C. and S. Deb, *Generation of optimal sequence of machining operations in setup planning by genetic algorithms.* Journal of Advanced Manufacturing Systems, 2012. **11**(01): p. 67-80.

[10] Yang, C.-L., S.-P. Chuang, and T.-S. Hsu, *A genetic algorithm for dynamic facility planning in job shop manufacturing.* The International Journal of Advanced Manufacturing Technology, 2011. **52**(1-4): p. 303-309.

[11] Chiung Moon et al, "An Efficient Genetic Algorithm for Traveling Salesman Problem" European J of Operational Research, Elsevier (2000)

[12] Pornsing, C. and A. Wattanasungsuit. *Genetic algorithm approach to the quality-related assembly line balancing problem*. in *Proceedings of the International MultiConference of Engineers and Computer Scientists*. 2008. Citeseer.

[13] Blum, C., *Ant colony optimization: Introduction and recent trends.* Physics of Life reviews, 2005. **2**(4): p. 353-373.

[14] Sormaz, D.N. and B. Khoshnevis, *Generation of alternative process plans in integrated manufacturing systems.* Journal of Intelligent Manufacturing, 2003. **14**(6): p. 509-526.

[15] Thierens, D. and D. Goldberg. *Elitist recombination: An integrated selection recombination GA*. in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. 1994. IEEE.

[16] Demuth H, Beale M. Neural Network Toolbox for Use with MATLAB. 2002.

[17] Umarani, R. and V. Selvi, *Particle Swarm Optimization: Evolution, Overview and Applications.* Int J of Engineering Science and Technology, 2010. **2**(7): p. 2802-2806.

[18] F. Villeneuve et al, "Feature state approach for operation sequence generation", Intelligent design & Manufacturing in Mechanical Engineering (1998), pp 93-102

[19] Farayi, M., *Reconfigurable Manufacturing Systems-What can Industrial Engineering and Management do?* Industrial Engineering & Management, 2012.

[20] Fisher, M.L., *Optimal solution of scheduling problems using Lagrange multipliers: Part I.* Operations Research, 1973. **21**(5): p. 1114-1127.

[21] Franco Busetti.“Simulated Annealing Overview”, (2003)

[22] Ma, G., Y. Zhang, and A. Nee, *A simulated annealing-based optimization algorithm for process planning*. International journal of production research, 2000. **38**(12): p. 2671-2687.

[23] Halevi, G. and R. Weill, *Principles of process planning: a logical approach*. 1995: Springer Science & Business Media.

[24] Singh, G. and G. Bartarya. *OPERATION SEQUENCING & MACHINING PARAMETERS SELECTION FOR ROTATIONAL COMPONENTS USING GENETIC ALGORITHM & EXPERT SYSTEM*. in *Proceeding of National Conference on Recent developments in Mech. Engg. TIET, Patiala vol. 1, pp218*. 2006.

[25] Venter, G., *Review of optimization techniques*. Encyclopedia of aerospace engineering, 2010.

[26] Michael, R.G. and S.J. David, *Computers and intractability: a guide to the theory of NP-completeness*. WH Freeman & Co., San Francisco, 1979.

[27] Di Caro, G., *Ant Colony Optimization and its application to adaptive routing in telecommunication networks*. 2004, PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium.

[28] Grassé, P.-P., *La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs*. Insectes sociaux, 1959. **6**(1): p. 41-80.

[29] ElMaraghy, H.A., *Reconfigurable process plans for responsive manufacturing systems*, in *Digital enterprise technology*. 2007, Springer. p. 35-44.

[30] Haupt, R.L. and S.E. Haupt, *Practical genetic algorithms*. 2004: John Wiley & Sons.

[31] Paris, J., L. Tautou-Guillaume, and H. Pierreval, *Dealing with design options in the optimization of manufacturing systems: an evolutionary approach*. International Journal of Production Research, 2001. **39**(6): p. 1081-1094.

[32] Jerald, J., et al., *Scheduling optimisation of flexible manufacturing systems using particle swarm optimisation algorithm.* The International Journal of Advanced Manufacturing Technology, 2005. **25**(9-10): p. 964-971.

[33] Hopfield, J.J. and D.W. Tank, *Computing with neural circuits- A model.* Science, 1986. **233**(4764): p. 625-633.

[34] Kerkesova Kristina. "Optimization Methods in Process Planning",5th International Multidisciplinary Conference,010 26 (2003)

[35] Niknam, K., A. Hasaninejad, and M. Arman, *Synthesis of some new bis-3, 4-dihydropyrimidin-2 (1H)-ones by using silica-supported tin chloride and titanium tetrachloride.* Chinese Chemical Letters, 2010. **21**(4): p. 399-402.

[36] Dorigo, M., *Optimization, learning and natural algorithms.* Ph. D. Thesis, Politecnico di Milano, Italy, 1992.

[37] M.K.Araffin, M.Badakhshian, S.B.Sulaiman, A.A.Faieza.. "Automated Guided Vehicles Scheduling by Fuzzy GA", Dept of Mechanical Engineering, UPM Darulehsan,43400, Malaysia (2003)

[38] Maghsud Solimanpur,Shahram Saeedi, IRaj Mahdavi. "Solving Cell Formation Problem in Cellular Manufacturing Using Ant-Colony-Based Optimization", Intl J Advance Manufacturing Technology (2010)

[39] Omar, M., A. Baharum, and Y.A. Hasan, *A JOB-SHOP SCHEDULING PROBLEM (JSSP) USING GENETIC ALGORITHM (GA).* 2006.

[40] Marco Dorigo, Thomas Stützle. Ant Colony Optimization. MIT Press, 2004.

[41] Martin, T.E., *Fitness costs of resource overlap among coexisting bird species.* Nature, 1996. **380**(6572): p. 338-340.

[42] Michael, P., *Scheduling, theory, algorithms, and systems.* Englewood Cli s, New Jersey, 1995.

[43] Ismail, N., et al., *Manufacturing process planning optimisation in reconfigurable multiple parts flow lines.* Journal of achievements in materials and manufacturing engineering, 2008. **31**(2): p. 671-677.

[44] Maheswaran, R., S. Ponnambalam, and N. Jawahar, *Hybrid heuristic algorithms for single machine total weighted tardiness scheduling problems.* International journal of intelligent systems technologies and applications, 2008. **4**(1): p. 34-56.

[45] Ahmad, N. and A. Anwarul Haque. *Optimization of process planning parameters for rotational components by genetic algorithms*. in *Proceedings of the 4th International Conference on Mechanical Engineering*. 2001.

[46] Bierwirth, C. and D.C. Mattfeld, *Production scheduling and rescheduling with genetic algorithms.* Evolutionary computation, 1999. **7**(1): p. 1-17.

[47] Pham, D. and D. Karaboga, *Intelligent optimisation techniques.* Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks, Springer, New York, 2000.

[48] Prabhu, P., et al., *An operations network generator for computer aided process planning.* Journal of Manufacturing Systems, 1990. **9**(4): p. 283-291.

[49] Rajabioun, R., *Cuckoo optimization algorithm.* Applied soft computing, 2011. **11**(8): p. 5508-5518.

[50] Rossi, A. and G. Dini, *Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method.* Robotics and Computer-Integrated Manufacturing, 2007. **23**(5): p. 503-516.

[51] Rumelhart, D.E., J.L. McClelland, and P.R. Group, *Parallel distributed processing: Explorations in the microstructures of cognition. Volume 1: Foundations.* MIT Press, Cambridge, MA, 1986. **2**: p. 560-567.

[52] Michalewicz, Z. and C.Z. Janikow. *Handling Constraints in Genetic Algorithms*. in *ICGA*. 1991

[53] Al-Mashari, M., A. Al-Mudimigh, and M. Zairi, *Enterprise resource planning: A taxonomy of critical factors.* European journal of operational research, 2003. **146**(2): p. 352-364.

[54]

[55] Nourali, S., N. Imanipour, and M.R. Shahriari, *A Mathematical Model for Integrated Process Planning and Scheduling in Flexible Assembly Job Shop Environment with Sequence Dependent Setup Times.*

[56] Zahid, T. and A.A. Baqai. *Multi-Criteria Optimization of Process Plans for Reconfigirable Manufacturing Systems: An Evolutionary Approach*. in *ASME 2013 International Mechanical Engineering Congress and Exposition*. 2013. American Society of Mechanical Engineers.

[57] Al-Mashari, M., A. Al-Mudimigh, and M. Zairi, *Enterprise resource planning: A taxonomy of critical factors.* European journal of operational research, 2003. **146**(2): p. 352-364.

[58] Nourali, S., N. Imanipour, and M.R. Shahriari, *A Mathematical Model for Integrated Process Planning and Scheduling in Flexible Assembly Job Shop Environment with Sequence Dependent Setup Times.*

[59] Singh, M., et al., *Cloning and characterization of a new theta-class glutathione-S-transferase (GST) gene, gst-3, from Drosophila melanogaster.* Gene, 2000. **247**(1): p. 167-173.

[60] Shabaka, A. and H.A. ElMaraghy, *Generation of machine configurations based on product features.* International Journal of Computer Integrated Manufacturing, 2007. **20**(4): p. 355-369.

[61] Burnwal, S. and S. Deb, *Scheduling optimization of flexible manufacturing system using cuckoo search-based approach.* The International Journal of Advanced Manufacturing Technology, 2013. **64**(5-8): p. 951-959.

[62] Chang, T.C. and R.A. Wysk, *An introduction to automated process planning systems*. 1984: Prentice Hall Professional Technical Reference.

[63] Tolio, T., et al., *SPECIES—Co-evolution of products, processes and production systems.* CIRP Annals-Manufacturing Technology, 2010. **59**(2): p. 672-693.

[64] Tartari Filho, S.C. and D.C. Donha, *AUTOMOBILE STOP-AND-GO CRUISE CONTROL SYSTEM TUNED BY GENETIC ALGORITHMS.* 2004.

[65] Vamsi Krishna, P., N. Shankar, and B. Surendra Babu, *FEATURE BASED MODELING AND AUTOMATED PROCESS PLAN GENERATION FOR TURNING COMPONENTS.* Advances in Production Engineering & Management, 2011. **6**(3).

[66] Oduguwa, V., A. Tiwari, and R. Roy, *Evolutionary computing in manufacturing industry: an overview of recent applications.* Applied Soft Computing, 2005. **5**(3): p. 281-299.

[67] Li, W., S. Ong, and A. Nee, *Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts.* International journal of production research, 2002. **40**(8): p. 1899-1922.Yang, X.-S. and S. Deb. *Cuckoo search via Lévy flights*. in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. 2009. IEEE.

[68] Yang, X. and N.-I.M. Algorithms, *Luniver Press.* Beckington, UK, 2008.

[69] Tu, X., et al. *Collaboratively scheduling to decrease inter-AS traffic in P2P live streaming*. in *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on*. 2008. IEEE.

[70] Yang, Y.-N., H. Parsaei, and H. Leep, *A prototype of a feature-based multiple-alternative process planning system with scheduling verification.* Computers & industrial engineering, 2001. **39**(1): p. 109-124.

[71] Koren, Y., et al., *Reconfigurable manufacturing systems.* CIRP Annals-Manufacturing Technology, 1999. **48**(2): p. 527-540.

[72] Yamada, Y., K. Ookoudo, and Y. Komura. *Layout optimization of manufacturing cells and allocation optimization of transport robots in reconfigurable manufacturing systems using particle swarm optimization*. in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. 2003. IEEE.

[73] Crama, Y., *Combinatorial optimization models for production scheduling in automated manufacturing systems.* European Journal of Operational Research, 1997. **99**(1): p. 136-153.

[74] Bi, Z., L. Wang, and S.Y. Lang, *Current status of reconfigurable assembly systems.* International Journal of Manufacturing Research, 2007. **2**(3): p. 303-328.

[75] Zhang, Y., W.B. Rossow, and P.W. Stackhouse, *Comparison of different global information sources used in surface radiative flux calculation: Radiative properties of the near-surface atmosphere.* Journal of Geophysical Research: Atmospheres (1984–2012), 2006. **111**(D13).

[76] Wen, Z., et al. *Use of approximate dynamic programming for production optimization*. in *SPE Reservoir Simulation Symposium*. 2011. Society of Petroleum Engineers.

[77] Bi, Z., *Revisiting system paradigms from the viewpoint of manufacturing sustainability.* Sustainability, 2011. **3**(9): p. 1323-1340.