DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

NEURAL NETWORK BASED OPTIMIZATION IN PRODUCTION SCHEDULING

by

Derya EREN AKYOL

September, 2006

İZMİR

NEURAL NETWORK BASED OPTIMIZATION IN PRODUCTION SCHEDULING

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in
Industrial Engineering, Industrial Engineering Program

by
Derya EREN AKYOL

September, 2006
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"NEURAL NETWORK BASED OPTIMIZATION IN PRODUCTION SCHEDULING"** completed by **DERYA EREN AKYOL** under supervision of **PROF. DR. G. MİRAÇ BAYHAN** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. G. Miraç BAYHAN

Supervisor

Prof. Dr. Semra TUNALI	Prof. Dr. Cüneyt GÜZELİŞ
Thesis Committee Member	Thesis Committee Member

Prof. Dr. Demir ASLAN	Assoc. Prof. Tijen ERTAY
Examining Committee Member	Examining Committee Member

Prof. Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

First and foremost I would like to express my deepest gratitude and thanks to my supervisor Prof. Dr. G. Miraç Bayhan for her support, friendly encouragement, guidance, and valuable advice throughout the progress of this dissertation.

I would like to express my sincere thanks to Prof. Dr. Cüneyt Güzeliş, a member of my dissertation committee, for generously sharing his time and valuable knowledge. I would also like to thank, another member of my committee, Prof. Dr. Semra Tunalı for her helpful comments and suggestions.

Special thanks go to Prof. Dr. Demir Aslan who introduced me the idea of using neural networks in scheduling.

I would like to thank Dr. Hatice Doğan for her helps when I needed.

I would like to express my thanks to all the professors and colleagues in the Department of Industrial Engineering for their support.

I would also like to thank my friends Dr. Gonca Tuncel, Özlem Uzun Araz, Ceyhun Araz for their encouragement and friendship.

Finally, I would like to express my indebtedness and many thanks to my parents, Metin Nami Eren and Müveddet Eren, my sister Oya, my brother Mete for their love, confidence, encouragement and endless support in my whole life. I would like to express my special thanks and deep appreciation to my husband Osman Altuğ Akyol for his love, support, companion and patience who always turned difficult times into joy with his smile and understanding.

Derya Eren Akyol

NEURAL NETWORK BASED OPTIMIZATION IN PRODUCTION SCHEDULING

ABSTRACT

Although a large number of approaches such as mathematical programming, dispatching rules, expert systems, and neighborhood search to the modeling and solution of scheduling problems have been reported in the literature, over the last decade, there has been an explosion of interest in using artificial neural networks (ANNs) for the solution of various scheduling problems.

The objective of this research is to utilize ANNs to deal with two different scheduling problems. The first problem considered is the classical identical parallel machine scheduling problem with makespan minimization. A dynamical gradient type neural network, which employs a penalty function approach with time varying coefficients, is proposed for the solution of the problem. Simulation outcomes of the proposed approach are compared with those of the longest processing time (LPT) rule and with the optimal solutions, for different sizes of scheduling problems. The second problem is the scheduling of an independent jobs set with sequence-dependent setups and distinct due dates on non-identical multi-machines to minimize the total weighted earliness and tardiness. The original mixed integer formulation of the problem is modified by adding one more constraint to the model to prevent the assignment of two jobs at the same time to the first position. For this problem, an interconnected neural network model which is composed of two maximum, three piecewise linear and one log-sigmoid neural networks all of which interact with each other is proposed. The proposed approach is tested on a scheduling problem and the results are compared with optimum results of the linear programming solver.

Keywords: Scheduling, Artificial neural networks, Dynamical neural network, Mixed integer formulation

ÜRETİM ÇİZELGELEMESİNDE YAPAY SİNİR AĞLARINA DAYALI EN İYİLEME

ÖZ

Çizelgeleme problemlerinin çözümü ve modellenmesi için literatürde matematiksel programlama, sevk etme kuralları, uzman sistemler ve komşu arama yaklaşımları gibi çok sayıda yaklaşım rapor edilse de yapay sinir ağları son on yılda çizelgeleme problemlerinin çözümü için kullanılma konusunda yoğun ilgi görmüştür.

Bu araştırmanın amacı, iki farklı çizelgeleme problemiyle uğraşmak için yapay sinir ağlarından yararlanmaktır. Dikkate alınan ilk problem maksimum tamamlanma süresini minimum yapma amacına sahip klasik özdeş makine çizelgeleme problemidir. Problemin çözümü için zamanla değişen katsayılara sahip ceza fonksiyonu yaklaşımı kullanan dinamik gradyan tip sinir ağı önerilmiştir. Önerilen yaklaşım kullanılarak farklı boyuttaki çizelgeleme problemleri için elde edilen benzetim sonuçları, en uzun işlem süresi kuralı sonuçları ve en iyi çözümler ile karşılaştırılmıştır. İkinci problem, ağırlıklandırılmış erken bitirme ve gecikme toplamını minimize etmek için özdeş olmayan makineler üzerinde sıra bağımlı hazırlık süreleri ve farklı teslim süreleri içeren bağımsız işler setinin çizelgelenmesidir. Problemin orijinal karışık tam sayı formülasyonu, ilk pozisyona aynı anda iki iş atanmasının önlenmesi için bir kısıt eklenerek değiştirilmiştir. Bu problem için birbirini etkileyen iki maksimum, 3 parçalı doğrusal, 1 tane log-sigmoid sinir ağından oluşan birbirine bağlı sinir ağı modeli önerilmiştir. Önerilen yaklaşım bir çizelgeleme probleminde test edilmiş ve sonuçlar doğrusal programlama çözümleyicisinin en iyi sonuçları ile karşılaştırılmıştır.

Anahtar sözcükler: Çizelgeleme, Yapay sinir ağları, Dinamik sinir ağı, Karışık tam sayı formülasyonu

CONTENTS

Page

THESIS EXAMINATION RESULT FORM ii
ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZ v

CHAPTER ONE – INTRODUCTION 1

1.1 Background and Motivation 1
1.2 Research Objective 2
1.3 Organization of the Thesis 4

CHAPTER TWO –OPTIMIZATION IN PRODUCTION SCHEDULING.......... 6

2.1 Basic Concepts and Performance Criteria 6

2.2 Optimization and Scheduling 10

2.3 Solution Approaches 11
2.3.1 Classical Methods 11
2.3.1.1 Mathematical Programming 12
2.3.1.2 Dynamic Programming 14
2.3.2 Constructive Heuristics 14
2.3.2.1 Dispatching Rules 14
2.3.2.2 Shifting Bottleneck Heuristic 16
2.3.3 Local Search Based Heuristics 16
2.3.3.1 Simulated Annealing 17
2.3.3.2 Tabu Search 17
2.3.3.3 Genetic Algorithms 18

2.3.3.4 Neural Networks 19

CHAPTER THREE – DYNAMICAL SYSTEMS FOR COMBINATORIAL OPTIMIZATION 21

3.1 Qualitative Analysis of Dynamical Systems 21
3.1.1 Liapunov's First Method for the Stability of an Equilibrium Point 22
3.1.2 La Salle's Invariance Theorem: Global Stability and Convergence 23

3.2 Dynamical Systems for Optimization 24
3.2.1 Gradient Based Systems 25
3.2.2 Hopfield Networks 26
3.2.2.1 Stability and convergence 28

CHAPTER FOUR – NEURAL NETWORKS IN PRODUCTION SCHEDULING........31

4.1 ANNs in Scheduling 31

4.2 Scheduling with Stand Alone Neural Networks 33
4.2.1 Hopfield Network and Its Extensions 33
4.2.2 Multilayer Perceptrons 40
4.2.3 Competitive Networks 45

4.3 Scheduling with Hybrid Approaches 48

4.4 Scheduling with Evolutionary Artificial Neural Networks........56

4.5 Summary and Future Research 60

CHAPTER FIVE – SOLUTION OF IDENTICAL PARALLEL MACHINE SCHEDULING PROBLEM USING DYNAMICAL GRADIENT NETWORKS…………………………………………………………….....69

5.1 Introduction 69

5.2 Problem Statement 72

5.3 Design of the Proposed Dynamical Gradient Network 73
5.3.1 The Network Architecture 73
5.3.2 The Energy Function 73
5.3.3 The Dynamics 77
5.3.4 Proof of Convergence 78
5.3.5 Selection of the Parameters 82
5.3.6 An example 85

5.4 Simulation Results 86

5.5 Conclusions 97

CHAPTER SIX – SOLUTION OF MULTI-MACHINE EARLINESS AND TARDINESS SCHEDULING PROBLEM USING AN INTERCONNECTED NEURAL NETWORK APPROACH…………………………………………….99

6.1 Introduction 99

6.2 Problem Statement and Formulation 102

6.3 Design of the Proposed Interconnected Neural Network 105
6.3.1 The Network Architecture 105
6.3.2 The Energy function 107
6.3.3 The Dynamics 112

6.3.4 Convergence 117
6.3.5 Selection of the Parameters 121

6.4 Simulation Results.. 123

6.5 Conclusions 129

CHAPTER SEVEN – RESULTS ANALYSIS AND DISCUSSION……………………..…………………………………………131

CHAPTER EIGHT – CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS…………………………………………………………………137

8.1 Conclusions 137

8.2 Future Research Directions 140

REFERENCES 142

CHAPTER ONE
INTRODUCTION

In this chapter, the background, motivation and objectives of this work are stated, and the organization of this dissertation is outlined.

1.1 Background and Motivation

Scheduling is one of the most important functions in manufacturing firms. It is the allocation of available production resources to tasks over time to meet some set of performance criteria. Typically the scheduling problem involves a set of jobs to be completed, where each job comprises a set of operations to be performed (Rodammer & White, 1989). These problems may arise in many fields of human activities such as scheduling of machines in a workshop, activities of a project, classes at a university, deliveries by a number of vehicles, flights of a fleet of airplanes, applications in a computer system, and patients in an emergency room.

The machine scheduling involves two kinds of decisions: sequencing (the order in which jobs are processed) and job-machine assignment. Machine scheduling problems are grouped into several classes, and parallel machine scheduling is one of these classes. In this thesis we will deal with this type of scheduling problem.

In the classical parallel machine scheduling problem, there are *n* jobs and *m* machines. Each job needs to be executed on one of the machines during a fixed processing time. Therefore, the aim is to find the schedule that optimizes a certain performance measure. Many real life problems can be modeled as parallel machine scheduling problems. Because, it is common to find more than one machine on production lines each kind carrying out the production tasks (Mokotoff, 2001).

It is known that the scheduling problem which belongs to a class of combinatorial optimization problems (COPs) is NP-hard. In the last decades, different solution methods such as mathematical programming, dispatching rules, expert systems, neighborhood search, and artificial neural networks (ANNs) have been proposed for modeling and solution of scheduling problems. After the success of Hopfield & Tank (1985), despite a vast amount of work existing in the literature, to find an efficient method for obtaining optimal solutions in polynomial time motivated the researchers to apply neural networks to scheduling problems and to compare their performance with other techniques'. The motivation behind the Hopfield & Tank neural network model was to take advantage of the great speed associated with the massively parallel computing capabilities of neural networks for fast solution of combinatorial optimization problems. Here, the motivation behind this research is to test the success of ANNs in solving parallel machine scheduling problems and to conclude about their performance. To the best of our knowledge, there are no previously published works tried to solve these NP-hard problems using neural networks.

1.2 Research Objective

In this thesis, we deal with two problems known to be NP hard. The first one is the identical parallel machine scheduling problem with makespan minimization. For the solution of this problem, we employ a dynamical gradient network. After the appropriate energy function is constructed by using a penalty function approach, the dynamics are defined by steepest gradient descent on the energy function. In this approach, the integral constraints on the decision variables are relaxed during computation and linear activation function is used to represent the continuous variable in the model. In order to overcome the tradeoff problem encountered in using the penalty function approach, a time varying penalty coefficient methodology is proposed to be used during simulation experiments. We analyze the impact that the initial conditions of the network have on the performance on 5 different data sets by running each data set 20 times for different sizes of jobs and machines.

The second one is the problem of scheduling a set of independent jobs with sequence-dependent setups and distinct due dates on non-identical multi-machines to minimize the total weighted earliness and tardiness. The original mixed integer formulation of the problem given by Zhu & Heady (2000) is modified by adding one more constraint to the model to prevent the assignment of two jobs at the same time to the first position. An interconnected neural network is developed to solve the problem. The proposed network is composed of two maximum neural networks, three piecewise linear networks and one log-sigmoid network all of which interact with each other. Since the model includes too many constraints to represent the problem, the energy function obtained by the penalty function approach incorporates many penalty terms corresponding to each constraint of the problem. Due to the tradeoff problem among the penalty terms, it becomes very difficult to find the values of the penalty parameters that result a feasible and a good solution. Some of the penalty terms are tried to be eliminated by the proposed network. Therefore, log-sigmoid and maximum networks are used to drop some of the penalty terms from the energy function. By this way, it is aimed to reduce the network complexity and to obtain a simplified energy function. Some of the binary constraints are satisfied using log-sigmoid networks, some binary constraints and assignment constraints are satisfied using maximum networks. A time varying penalty coefficient methodology is also proposed to be used during simulation experiments to overcome the tradeoff problem encountered in using the penalty function approach, and the proposed approach is tested on a parallel machine scheduling problem.

The objectives of this thesis are listed below.

- To present a detailed evolutionary path of ANNs in production scheduling, review the current research literature, classify the approaches according to their architectures and to discuss several future research directions.

- To present a literature review on identical parallel machine scheduling and earliness and tardiness scheduling models.

- To propose and evaluate dynamical gradient type neural network models for solving two parallel machine scheduling problems namely the identical parallel machine scheduling with makespan minimization, and the scheduling a set of independent jobs with sequence-dependent setups and distinct due dates on non-identical multi-machines to minimize the total weighted earliness and tardiness.

- To propose a penalty determination method in which the penalty parameters vary with time, to overcome the tradeoff problem, the biggest drawback of the penalty function approach used to construct the energy function of the proposed gradient networks.

- To investigate the methods of reducing the complexity of the proposed gradient network which is originally described by an energy function including many penalty parameters.

- To provide the experiment results comparing the proposed neural network method with dispatching rules and/or with the optimum solutions obtained by a linear programming solver.

1.3 Organization of the Thesis

The organization of this dissertation is as follows.

Chapter 2 is an introduction to production scheduling. Some definitions and concepts in scheduling theory are provided along with an overview of solution approaches used for solving scheduling problems.

In Chapter 3, we consider a general continuous time dynamical system defined by state equations and present the conditions for the global existence and uniqueness of solutions. The definitions of equilibrium point, stable equilibrium point, asymptotic stable equilibrium point are given followed by the explanation of stability analysis of an equilibrium. La Salle’s invariance theorem which defines the conditions for convergence of the solution are given and gradient based systems including Hopfield networks are also described in this chapter.

Chapter 4 presents an extensive review of the different neural network implementations in solving the scheduling problems. A general overview of how neural networks are used in solving scheduling problems and what makes them appropriate tools for solving scheduling problems are given. Adopting a chronological approach, the approaches are classified under three main categories: (1) classical neural network approaches, (2) hybrid approaches, and (3) evolutionary neural network approaches. The advantages, disadvantages and suitability of approaches in each category for solving scheduling problems are discussed and possible future research directions are given.

In Chapter 5, the problem of minimizing the maximum completion time (makespan) of jobs on identical parallel machines is introduced and the relevant literature review is given. The proposed dynamical gradient network is explained and the convergence of the network is proved. The simulation results are given for different sizes of problems on different data sets.

In Chapter 6, the problem of scheduling a set of independent jobs including sequence dependent setup times, on non-identical multiple machines to minimize the total weighted earliness and tardiness is studied and the relevant studies in the literature are reviewed. The proposed interconnected network is presented and the convergence of the network is discussed by extending La Salle’s invariance theorem to systems with discontinuous right-hand sides. The proposed approach is illustrated through the case of 8 jobs to be processed on 3 machines in a JIT manufacturing environment.

In Chapter 7, in-depth discussion and analysis of our results is given.

Chapter 8 contains the concluding remarks of this research and identifies future research directions.

CHAPTER TWO
OPTIMIZATION IN PRODUCTION SCHEDULING

In this chapter, we give an introduction to the area of scheduling, present some terminology and concepts of scheduling theory with an overview of the optimization techniques and heuristics used in solving scheduling problems.

2.1 Basic Concepts and Performance Criteria

Scheduling is concerned with the allocation of limited resources to perform a collection of given tasks. The resources may be machines in a workshop, runways at an airport, crews at a construction site, and processing units in a computing environment. The tasks may be operations in a production process, take-offs and landings at an airport, stages in a construction project, execution of computer programs (Pinedo, 1995). In general, these tasks have to be accomplished with the goal of minimizing or maximizing an objective. Each task may have different priorities, different due dates and different arrival times, and the objectives may take many forms such as the minimization of the total time required to complete the processing of all jobs, minimization of the total tardiness, etc.

Generating a production schedule involves selecting a sequence of operations that will result in the completion of a job, designating the resources needed to execute each operation in the routing and assigning the times at which each operation in the routing will start and finish execution (Rodammer & White, 1988). It is a decision-making process and exists in most manufacturing and production systems as well as in most information-processing environments. It also exists in transportation and distribution settings and in other types of service industries (Pinedo, 1995). Extensive reviews on scheduling can be found in (Rodammer & White, 1988; Reklaitis, 1992; Zenter & Pekny, 1994). In a recent paper, Shah (1998) and Chen, Potts, & Woeginger (1998) also provided a detailed review on the current status of single and multisite scheduling and planning problems.

The importance of scheduling comes from its wide applicability in manufacturing as well as in services. It is concerned as one of the important application areas of optimization (Pardalos & Resende, 2002). The general scheduling problem includes a set of *n* jobs or tasks J={J_1, J_2, ..., J_n} and a set of *m* machines or processors M={M_1, M_2,…,M_m} and is usually characterized by the machine environment, the objective function, the processing restrictions, and the constraints. Usually, the three field notation $\alpha|\beta|\gamma$ of Graham, Lawler, Lenstra, & Rinnooy Kan (1979) is used to describe scheduling problems where α denotes the machine environment, β specifies the job characteristics and γ corresponds to the performance measures used (Pardalos & Resende, 2002). The two general rules that must be followed in scheduling theory (Eiselt & Sandblom, 2004) are as follows:

- Each job is to be processed by at most one machine at a time.
- Each machine is capable of processing at most one task at a time.

The processing of the jobs may be subject to several types of restrictions and constraints. For example, while preemption of jobs may be possible in some environments, some may not involve preemptions. A schedule is *non-preemptive* if each job runs uninterrupted on one machine from start to finish. In a *preemptive* schedule, a job may be interrupted or may switch other machines at any time. There may also be routing constraints that specify the route each job has to follow through the shop (Pardalos & Resende, 2002).

In general, scheduling problems can be grouped into three main classes according to machine environments.

- Single machine scheduling problems,
- Parallel machine scheduling problems,
- Shop scheduling problems
  -Flow shop scheduling problems
  -Job shop scheduling problems
  -Open shop scheduling problems.

In single machine scheduling models, as its name implies, there is only one machine and the routes consist of only one operation performed on this machine (Artiba & Elmaghraby, 1997). The parallel machine scheduling problem is an important generalization of the single machine problem which can be a subproblem in many complex multi-machine problems (Hax & Candea, 1984). In parallel machine environments, each task can be processed by any machine. There are several machines which are distinguished depending on their speeds. If all the machines have equal speeds, in other words if they are able to process the tasks with processing times independent of the machines, they are called *identical* machines. Therefore, processing of job i on any machine j takes the same time, i.e. $p_{ij} \equiv p_i$. In the case of uniform machines, the machines differ in their speeds but the speed of each machine is constant and does not depend on the task it processes. In this case, the processing times of a job on each machine may differ by speed factors, i.e. $p_{ij} = p_i/s_j$, where s_j reflects the speed of machine j, and p_i is called the standard processing time of job i (usually on the slowest machine). If the processing time of job i varies between machines in an arbitrary fashion, they are called *unrelated* machines. For a detailed literature review on parallel machine scheduling problems, one can refer to (Cheng & Sin, 1990; Mokotoff, 2001).

In scheduling models, each job is characterized by some parameters that are given below.

- Processing time (p_{ij}): It is the amount of time required by job i on machine j to complete its processing.

-Arrival time or ready time or release date (r_j): This is the time at which job j is available for processing.

-Due date (d_i): It is the time at which job i should be completed.

-A weight or priority (w_i): The weight of a job designates its importance relative to other jobs present in the system. It is used to express the priority of job i.

For any given schedule, the following primary output measures which are used to determine other output measures can be calculated for each job *i*.

Completion time C_i: It is the time by which job *i* completes its processing.

Flowtime F_i: It is the amount of time job *i* spends in the system between its arrival and its completion.

$F_i=C_i-r_i$

Lateness L_i: Lateness is expressed as the amount of time by which the completion of job *i* exceeds its due date.

$L_i=C_i-d_i$

Tardiness T_i: Tardiness can be calculated with respect to the lateness. The tardiness is equal to the lateness of job *i* if lateness is positive, and the tardiness is equal to zero if the lateness of job *i* is not positive.

$T_i = \max\{C_i-d_i, 0\} = \max\{L_i, 0\}$.

Earliness E_i: Earliness is the negative of the lateness of job *i* if the lateness is negative, and the earliness is zero if the lateness is positive.

$E_i=max\{0, -L_i\}$.

To evaluate any schedule, the following performance measures which are also known as objective functions or performance criteria are used. They are generated from the primary output measures.

Makespan C_{max}: It refers to the time it takes to complete all *n* jobs, that is the overall completion time, $C_{max} = \max(C_1, \ldots, C_n)$.

Mean flow time $\overline{F} = \frac{1}{n}\sum_{j=1}^{n} F_j$

Weighted flow time $F_w = \sum_{j=1}^{n} w_j F_j$

Mean weighted flow time $\overline{F}_w = \dfrac{\sum_{j=1}^{n} w_j F_j}{\sum_{j=1}^{n} w_j}$

Maximum flowtime $F_{max} = \max_j\{F_j\}$

Weighted lateness $L_W = \sum_{j=1}^{n} w_j L_j$

Maximum lateness $L_{max} = \max_j\{L_j\}$

Weighted tardiness $T_w = \sum_{j=1}^{n} w_j T_j$

Maximum tardiness $T_{max} = \max_j\{T_j\}$

Mean tardiness $\overline{T} = \dfrac{1}{n}\sum_{j=1}^{n} T_j$

Mean weighted tardiness $\overline{T}_w = \dfrac{\sum_{j=1}^{n} w_j T_j}{\sum_{j=1}^{n} w_j}$

Weighted number of tardy jobs $N_w = \sum_{j=1}^{n} w_j \delta(T_j)$

where $\delta(T_j)=1$ if $T_j>0$, and 0 otherwise.

Total weighted earliness and tardiness $ET_w = \sum_{j=1}^{n} (w_{E_j} E_j + w_{T_j} T_j)$

2.2 Optimization and Scheduling

Optimization is the process of maximizing or minimizing a desired objective function while satisfying the prevailing constraints (Belegundu & Chandrupatla, 1999), and optimization problems in which the possible solutions form a finite set are called combinatorial optimization problems. In other words, they are a general class of optimization problems restricted to a finite and discrete solution space. Combinatorial optimization problems in scheduling are the efficient allocation of limited resources to meet desired objectives when the values of some or all of the variables are restricted to be integer. Constraints on basic resources such as labor, supplies or capital, restrict the possible alternatives that are considered to be feasible.

Scheduling problems form an important class of combinatorial optimization problems. They execute a given set of jobs by utilizing several machines and other resources subject to certain constraints such as priority and deadline constraints. They are generally NP-hard, that is, it is probably impossible to secure optimal solutions using fast algorithms (i.e. algorithms that run in polynomial time in the size of the problem), though some problems, such as the single machine scheduling with flow time minimization problem, are easy in the sense that they are solvable to optimality by fast algorithms.

The amount of computational requirement for most combinatorial optimization problems in scheduling is the factorial of the problem size. Hence, complete enumeration to reach the optimal solution is not always possible as the number of possible solutions increases as the factorial of the problem size (Cheung, 1994). Therefore, in general, an approximation schema is constructed for the whole problem without any guarantee of optimality. Besides exact methods that solve problems to optimality, several methods and techniques such as constructive methods and local search methods may be used to build an approximation schema.

2.3 Solution Approaches

2.3.1 Classical Methods

Classical methods use an appropriate mathematical description of the scheduling problem. They do not try to investigate all of the possible feasible solutions, which would be practically impossible, and they reduce the search space and the CPU time required to obtain a solution, while satisfying the constraints.

2.3.1.1 Mathematical Programming

In mathematical programming, scheduling decisions are modeled using integer and continuous variables, and the scheduling problem is represented as an optimization problem in which a mathematical function has to be minimized or maximized subject to some linear and non-linear algebraic constraints. If the objective function is linear and the constraints are a combination of linear equalities or inequalities, the problem is called a linear programming problem. In a linear programming problem, the decision variables involved in the problem are also nonnegative (i.e., positive or zero). The simplex method developed by George Dantzig in 1947 is an iterative procedure for generating and examining different extreme points of a linear program, each one improving the current value of the objective function until an optimum is found. If some of the variables in a linear programming model are required to have integer values, this model is referred to as *mixed* integer programming (MIP) and if all the variables are integers, it is called a *pure* integer programming problem. The word integer programming (IP) often has reference to both pure integer and mixed integer programming problems. Although IP provides a lot of flexibility in formulating scheduling problems, integer variables make an optimization problem non-convex, and consequently far more difficult to solve. Memory and solution time may rise exponentially as more integer variables are added. Even with highly sophisticated algorithms and modern supercomputers, there are models of just a few hundred integer variables that have never been solved to optimality. One can refer to Nemhauser & Wolsey (1988) for more information about integer and combinatorial optimization. A recent detailed review on integer modeling techniques for scheduling has been presented by Blazewicz, Dror, & Weglarz (1991) and Pinto & Grossmann (1998).

The classical approach for solving IP problems is a branch-and bound algorithm. Branch-and-Bound is a methodology developed for solving many types of combinatorial optimization problems. The actual implementation of a branch-and-bound algorithm is typically viewed as a tree search where the problem at the root node of the tree is the original IP. It is based on the idea of enumerating all feasible solutions and consists of two procedures; branching and bounding. By branching, the problem is iteratively partitioned into subproblems of the original IP and is represented by a node in its corresponding branch and bound tree. In the bounding procedure, a lower bound is calculated on the optimal solution of a given subproblem to shorten the enumeration process. Each subproblem can be partitioned in a similar manner. When the lower bound given in the node is greater than or equal to the best feasible solution, the node and its subproblems are not explored further and its branch is said to be fathomed. In this procedure, certain schedules or classes of schedules are discarded by showing that the values of the objective obtained with schedules from this class are larger than a provable lower bound which is greater than or equal to the value of the objective of a schedule obtained earlier (Pinedo, 1995). When a better feasible solution is obtained, it becomes the current best feasible solution. If a node has not been eliminated, and its subproblems have not been generated, it is called active node, and the branch-and-bound algorithm stops when no active nodes remain.

Cutting plane algorithms are also known among other methods for solving IP problems. These algorithms were first proposed by Gomory (1958, 1963), and in recent years have been proven to be computationally useful especially when combined with a branch and bound algorithm in a branch and cut framework. These methods work by solving a sequence of linear programming relaxations of the integer programming problem. The relaxations are gradually improved to give better approximations to the IP problem, at least in the neighborhood of the optimal solution. For hard instances that can not be solved to optimality, cutting plane algorithms can produce approximations to the optimal solution in moderate computation times. A survey of applications of cutting plane methods can be found in Jünger, Reinelt, & Thienel (1995).

2.3.1.2 Dynamic Programming

Dynamic programming introduced by Bellman (1957) is a mathematical technique used for solving small and medium sized problems that are pseudo-polynomially solvable. It is an enumeration method that uses a divide and conquer approach, and finds optimal solutions to subproblems. Then, according to the principle of optimality, it solves the problem recursively. Since it performs an intelligent enumeration of all feasible points, it resembles the branch-and bound method.

2.3.2 Constructive Heuristics

Instead of trying to improve an initial solution using moves in a given neighborhood, the constructive procedures build a schedule from scratch by inserting unscheduled operations into a partial schedule until the schedule is complete.

2.3.2.1 Dispatching Rules

In the scheduling literature, terms such as scheduling rule, dispatching rule or priority rule are often used synonymously. A dispatching rule prioritizes all the jobs that are waiting for processing on a machine. In other words, a dispatching rule is used to determine the next job waiting in front of a machine when the machine becomes available. The prioritization scheme may take into account the jobs' attributes, the machines' attributes as well as the current time. Whenever a machine has been freed, a dispatching rule selects among the jobs waiting, the job with the highest priority. For several decades, many different rules have been developed and studied in the literature for different scheduling objectives. Some of the dispatching rules commonly used in scheduling are as follows:

First in First Out (FIFO) rule: With this rule, the job with the earliest release date is chosen first for processing. It minimizes the maximum flowtime and the variation in the waiting times of the jobs at a machine.

Shortest Processing Time (SPT) rule: Using this rule, the job with the shortest processing time is chosen fist for processing. This rule provides an optimal schedule for the mean flow time criterion.

Longest Processing Time (LPT) rule: This rule arranges jobs in descending order of processing times, such that $p_1 \geq p_2 \geq \ldots \geq p_n$ and assigns the first *m* jobs to the *m* machines at time $t = 0$. After that, whenever a machine becomes available, the largest unscheduled job is loaded on the machine. This rule tends to minimize makespan in parallel machine scheduling environments.

The earliest due date (EDD) rule: Whenever a machine is freed, the job with the earliest due date is selected to be processed next. It tends to minimize the maximum lateness among the jobs waiting for processing.

The minimum slack (MS) rule: Whenever a machine is freed at time *t*, the remaining slack of each job at that time, defined as max $(d_j\text{-}p_j\text{-}t,0)$ is found and the job with the minimum slack is scheduled next. This rule tends to minimize due date based objectives.

The weighted shortest processing time first (WSPT) rule: According to this rule, the jobs are ordered in decreasing order of w_j / p_j. It tends to minimize the weighted sum of the completion times expressed as $\sum_{j=1}^{n} w_j C_j$.

The main advantage of dispatching rules is that they are easy to understand, easy to apply, and require relatively little computer time. Their primary disadvantage is that they can’t guarantee an optimal solution. Panwalkar & Iskander (1977) and Blackstone, Philips, & Hogg (1982) provide a good survey on dispatching rules.

2.3.2.2 Shifting Bottleneck Heuristic

Shifting Bottleneck Heuristic is a decomposition method which was first proposed by Adams, Balas, & Zawack (1988) and designed for the minimization of the makespan in a job shop. Using this method, the job shop problem is decomposed into single machine subproblems and the machines are scheduled iteratively one after another. In each iteration, the next machine to be included is the one causing the largest maximal lateness known to be the bottleneck machine and the schedule is fixed for the current bottleneck machine. This method also involves a reoptimization process, that is each time a new machine is scheduled, each of the machines previously scheduled is considered again as an unscheduled machine by deleting the disjunctive arcs that had been fixed before. The corresponding subproblem is reformulated by recomputing the data necessary and the machine is then rescheduled using the same branch-and bound algorithm.

Applications of shifting bottleneck heuristic for problems with other objectives, Lmax, $\sum_{j=1}^{n} w_j T_j$,can be found in Ovacik & Uzsoy (1997), Pinedo & Singer (1997), and Yang, Kreipl, & Pinedo (1997).

2.3.3 Local Search Based Heuristics

Local search is one of the basic methods used to find approximate solutions for hard combinatorial optimization problems, known to be NP-hard. Since computing exact optimal solutions is considered to be computationally intractable for some combinatorial optimization problems, approximate (or heuristic) algorithms are developed. One of the important tools is the local search that starts from an initial solution *x* and repeatedly replaces it with a better solution in its neighborhood *N(x)* until no better solution is found in *N(x)*; where *N(x)* is a set of solutions obtainable from *x* by slight perturbations. Local search methods do not guarantee an optimal solution and usually attempt to find a better schedule than the current one in the neighborhood of the current one (Pardalos & Resende, 2002).

Local search consists in moving from a solution to another one in the neighborhood according to some definite rules. The sequence of solutions can be called a *trajectory* in the solution space. They depend heavily on initial solutions and the neighbourhood generation mechanisms (Madureira, 1999).

Some of the principal local search based strategies are Tabu Search, Simulated Annealing, Genetic Algorithms and Neural Networks.

2.3.3.1 Simulated Annealing

Simulated annealing proposed by Kirkpatrick, Gelatt, & Vecchi (1983) belongs to a class of local search algorithms that are known as threshold algorithms. The idea of simulated annealing stems from the physical annealing process. Annealing is a physical process in which a solid is heated up to a maximum value at which all particles of the solid randomly arrange themselves in the liquid phase, followed by a process of cooling the solid slowly, until the solid reaches a stable state. It is a stochastic approach that tries to find a good solution to an optimization problem by trying random variations of the current solution. A worse solution is accepted as the new solution with a probability that decreases as the computation proceeds. In other words, not only downhill moves but uphill moves are also accepted probabilistically. The slower the cooling schedule, the more likely the algorithm is to find an optimal or near optimal solution. (Artiba & Elmaghraby, 1997).

Simulated annealing has received considerable attention in the last years and has been applied to solve a variety of problems including scheduling (Osman & Potts, 1989; Van Laarhoven, Aarts, & Lenstra, 1992; Radhakrishnan & Ventura 2000; Eglese, 1990).

2.3.3.2 Tabu Search

Tabu Search is a local search method guided by the use of adaptive memory structures. It was first presented in its present form by Glover (1989, 1990) as a deterministic alternative to simulated annealing and has been successfully applied to a variety of combinatorial optimization problems including scheduling. The basic idea of the method is to explore the solution space by a sequence of moves made from one solution to another better one, and it is possible to leave local optima by accepting a degradation of the criterion value but without using previous visited solutions which are stored in a list of forbidden or 'tabu' solutions (Artiba & Elmaghraby, 1997). By this way, the cycling is prevented and search diversification is encouraged ((Pardalos & Resende, 2002). The tabu list is dynamically updated during the execution of the algorithm and the solutions defined by the tabu list are not acceptable in the next iterations. However, a solution on the tabu list may be accepted if its quality is in some sense good enough, in which case it is said to attain a certain aspiration level (Aarts & Lenstra, 1997).

2.3.3.3 Genetic Algorithms

Genetic algorithms (GAs) introduced by John Holland (1975) are known as search algorithms, which explore a solution space and try to mimic the natural evolution process. Works in the area of GAs were summarized in Schaffer, Whitley, & Eshelman (1992), Goldberg (1989), Schaffer (1989), Belew & Booker (1991) and Forrest (1993). They proved to be very useful in many applications such as mathematics, robotics, physics, and in many optimization problems including scheduling. One can refer to Cheng, Gen, & Tsujimura (1996) and Cheng *et al.* (1999) for a detailed survey on GA applications in job shop scheduling which is one of the main scheduling fields GAs are applied to. Other examples of GA applications can be found in Ahmed, Kwok, Ahmad, & Dhodhi (2001), Hou, Ansari, & Ren (1994), Theys, Braun, Siegal, Maciejewski, & Kwok (2001) and Zomaya, Lee, & Olariu (2001).

GA starts with the creation of an initial population of possible solutions to the problem called individuals or chromosomes, and the genes within the chromosomes determine the individual features of the child. Each chromosome is associated with a fitness value, which represents the probability of a chromosome being selected to be a parent. From the individual population, a new population is generated using one of the specific operators such as reproduction, crossover or mutation. By the reproduction operator, the solutions in the old population are copied to the next population with a probability depending on the fitness of the solution which corresponds to the value of the objective function for that solution. Using the crossover operator, new solutions are generated from pairs of individuals, and by mutation one or more of the genes in a chromosome are altered in a random way which helps the GA to explore a new, perhaps a better feasible region than the previously considered. The process is repeated until some stopping rule is satisfied and the individual with the most favorable fitness is the solution to the problem.

2.3.3.4 Neural Networks

ANNs were originally designed for simulating the brain behaviour. They have emerged as efficient approaches in a variety of engineering applications where problems are difficult to formulate or hardly defined. They are computational structures that implement simplified models of biological processes, and are preferred for their robustness, massive parallelism and ability to learn. In metaheuristics literature, neural networks are put into local-search based metaheuristics category. The reason is their iterative master process characteristic, that is, they guide and modify the operations of subordinate heuristics to efficiently produce high quality solutions, and provide decision makers with fast and robust tools for obtaining high quality solutions in reasonable computation times to many real life problems.

From a modeling viewpoint, they are mathematical representations of biological nervous systems that can carry out complex cognitive and computational tasks. They are composed of many nonlinear interconnected processing elements that are analogous to neurons, and connected via weights that are analogous to synapses. The modern age of neurocomputing started with the work of McCulloch & Pitts (1943) in which the first mathematical model of a single biological neuron was presented. Although McCulloch and Pitts' study showed that simple type of neural networks were able to learn arithmetic or logical functions, ANN algorithms have been successful enough for many applications in the mid 1980s (Potvin & Smith, 2003). The field attracted the attention of many researchers from different disciplines such as engineering, physics, mathematics, computer science and medicine. In recent years, ANNs have become popular in various real world applications including prediction and forecasting, function approximation, clustering, speech recognition and synthesis, pattern recognition and classification, and many others. Applications of ANNs to manufacturing scheduling (for detailed survey see Sabuncuoglu, 1998) are in accordance with using ANNs as a highly parallel model for general-purpose computing and then applying them for different combinatorial optimization problems (for detailed survey see Looi, 1992; Smith, 1999).

In the literature, although a large number of approaches such as mathematical programming, dispatching rules, expert systems, and neighborhood search to the modeling and solution of scheduling problems have been reported, over the last decades, there has been an explosion of interest in using ANNs. Certainly, as the ANNs evolve, solutions of scheduling problems will make progress. In this thesis, Hopfield type dynamical gradient networks are used to solve the problems under consideration. While details of the dynamical neural networks usage in solving optimization problems will be given in the following chapter, a comprehensive review on the applications of ANNs in scheduling will be given in Chapter 4.

CHAPTER THREE
DYNAMICAL SYSTEMS FOR COMBINATORIAL OPTIMIZATION

In this thesis, dynamical gradient based neural networks are proposed for solving the problems under consideration and gradient systems are constructed utilizing the concepts developed for stability analysis of dynamical systems. In this chapter, qualitative analysis of dynamical systems followed by the explanation of dynamical systems for optimization including gradient networks and Hopfield networks is given parallel to the studies of Desoer (1970), Hirch & Smale (1974), Aggarwal & Vidyasagar (1977), Chua & Wang (1978), Hopfield (1982), Hopfield (1984), Hopfield & Tank (1985), Cakir (2002) and Dogan (2004).

3.1 Qualitative Analysis of Dynamical Systems

Suppose that a dynamical system is given as follows.

$$\dot{x}(t) = f(x(t),t),\ x(t_0) = x_0,\ f(.,.) : R^n \times R \to R^n \qquad (3.1)$$

The following theorem explains the existence and uniqueness of a global solution of (3.1).

Theorem 3.1 (Desoer, 1970) suppose the following conditions hold.

i) $S \subset R^+$ contains at most a finite number of points per unit interval,

ii) For each x Є R^n, f(x,.) is continuous at $t \notin S$

iii) For each $t_i \in S$, f(x,.) has a finite left-hand and right hand limits at t=t_i

iv) f(.,t) : $R^n \to R^n$ satisfies a global Lipschitz condition. That is, there exists a piecewise continuous function k(.): $R^+ \to R^+$ such that $\|f(x_1,t) - f(x_2,t)\| \le k(t)\|x_1 - x_2\|$ for all t Є R$^+$ and for all points x_1, x_2 Є R^n.

Then, for each x_0 Є R^n and t_0 Є R$^+$, there exists unique continuous solution $\phi(.;t_0,x_0) : R^+ \to R^n$ satisfying that state equation and the initial condition:

(a) $\dot{\phi}(t,t_0,x_0) = f(\phi(t,t_0,x_0),t)$

(b) $\dot{\phi}(t_0,t_0,x_0) = x_0$ for all $t_0 \in \mathbb{R}^+$ and $t \notin S$

3.1.1 Liapunov's First Method for the Stability of an Equilibrium Point

Liapunov stability is concerned with the behavior of solution when its initial state is near the equilibrium. After sufficiently long time, if the solution remains near the equilibrium point then we can say that the equilibrium is stable.

For the system given in (3.1) with t € [t_0, ∞] formulation of Liapunov stability analysis of an equilibrium point is given as follows

Definition 3.1 Equilibrium Point

A solution $\phi(t;t_0,x_e)$ is called an equilibrium point if it is constant for all times and satisfies the state equations:

i) $\phi(t;t_0,x_e) = x_e \quad \forall t \in [t_0,\infty]$

ii) $\dot{\phi}(t;t_0,x_e) = f(\phi(t;t_0,x_e),t) = 0$

Definition 3.2 Stability of an equilibrium point (Aggarwal and Vidyasagar, 1977)

An equilibrium state x_e of a dynamical system given by (3.1) is stable in the sense of Liapunov if for every real number ε > 0 there exists a real number δ(ε, t_0) > 0 such that $\|x_0 - x_e\| \le \delta$ implies $\|\phi(t;t_0,x_0) - x_e\| \le \varepsilon$ for all $t \ge t_0$ where $\phi(t;t_0,x_0)$ is a solution at time *t*.

If the solutions initiated near the equilibrium point converge to the equilibrium, we call this equilibrium as asymptotically stable.

Definition 3.3 Asymptotic stability of an equilibrium

An equilibrium state x_e of a dynamical system given in (3.1) is asymptotically stable if it is stable and every solution starting in a sufficiently small neighborhood of x_e converges to x_e as $t \to \infty$.

3.1.2 La Salle's Invariance Theorem: Global Stability and Convergence

In some systems, local stability of an equilibrium may not be enough. No matter how large the perturbation is, it is desired that the system returns to its unique equilibrium state. This property is called as global asymptotic stability. The definitions of global asymptotic stability and convergence will be given below.

Definition 3.4 Convergency

A system $\dot{x} = f(x,t)$ with $x \in \mathrm{R}^n$ is called convergent if any trajectory, namely solution $\phi(t;t_0,x_0)$ ends at one of the equilibria depending on the initial state x_0. According to the definition, convergent systems don't possess oscillatory or other exotic dynamics but possess constant steady state behavior only.

Definition 3.5 Global Asymptotic Stability

An equilibrium point x_e of a system $\dot{x} = f(x,t)$ $x \in \mathrm{R}^n$ is called globally asymptotically stable if the system is convergent and the equilibrium point is unique. The theorem of La Salle given below states that under certain conditions any solution tends to an invariant set which might consist of a unique equilibrium point, a set of equilibria, periodic solutions or other complicated solutions. It should be mentioned that La Salle's invariance theorem works with nonincreasing Lyapunov functions for deciding asymptotic stability of an equilibrium point. Hence, it constitutes a useful extension of Liapunov asymptotic stability result valid only for strictly decreasing Lyapunov functions (Dogan, 2004).

Theorem 3.2 La Salle's invariance (Aggarwal and Vidyasagar, 1977)

Consider a system $\dot{x} = f(x)$ with f(.) Є C^1. If there exists a continuously differentiable Lyapunov function V (.): $R^n \to R^1$ such that

i) the set $\Omega_r = \{x \in R^n | V(x) \le r\}$ is bounded for some r > 0,

ii) V(.) is bounded below over such a set Ω_r, and

iii) $\dot{V} \le 0 \quad \forall X \in \Omega_r$, then any solution *x(t,x_0,0),* starting from *x_0=x(0)*$\in \Omega_r$ tends to the largest invariant set contained in S: = $\{x \in \Omega_r | \dot{V}(x) = 0\} \subset \Omega_r$.

The largest invariant set expressed in Theorem 3.2 consists of equilibrium points if the conditions of Theorem 3.3 are satisfied.

Theorem 3.3 (Chua & Wang, 1978)
The autonomous system $\dot{x} = f(x)$ with f(.) Є C^1 is convergent, namely the invariant set which the trajectories tend to is made up of the equilibrium points if

i) the solutions of the system are bounded,

ii) there exists a continuously differentiable Liapunov function V(.) such that $\dot{V} \le 0 \quad \forall X \in R^n$ along trajectories. Furthermore, $\dot{V}$ vanishes at equilibrium points only.

3.2 Dynamical Systems for Optimization

An optimization problem can be solved by using an analog network as a dynamical solver whose equilibrium points correspond to the solutions of the optimization problem. Although most of the combinatorial problems are computationally intractable, an appropriate dynamical system may be used to find a satisfactory solution in reasonable time. One of the methods used in designing dynamical solvers is the construction of a set of differential equations such that their equilibrium points correspond to the desired solutions, and then finding an appropriate Liapunov function such that all trajectories of the system converge to equilibrium points. Commonly used approach in designing dynamical solvers is based on gradient systems that use steepest descent method as a basis. In these methods, the minimization problem is transformed into an associated system of first order differential equations as follows.

$$\dot{x} = \mu(x)\nabla E(x) \tag{3.2}$$

where, E(x) is the cost function of the unconstrained optimization problem. Dynamical system $\dot{x} = -\nabla E(x)$ obtained for $\mu(x) = I$ is called a gradient system (Cakir, 2002).

3.2.1 Gradient Based Systems

A gradient system on an open set U $\subset$ R^n is a dynamical system of the form

$$\dot{x} = -\nabla V(x) \tag{3.3}$$

where V(.): $U \to R$ is usually a C^2 function. These systems are convergent if their solutions are bounded. To prove this, Theorem 3.3 can be considered, and then the derivative $\dot{V} : U \to R$ of V along the trajectories of the system given by (3.3) can be written as below.

$$\dot{V}(x) = \frac{\partial V(x)}{\partial x}\frac{dx}{dt} = [\nabla V]^T \frac{d(x)}{dt} = -[\nabla V]^T[\nabla V] \leq 0 \tag{3.4}$$

It is known that for an unconstrained optimization problem, if $x*$ is a local minimum point of f(.), then $\nabla f(x*) = 0$, together with this information, it can be concluded that any equilibrium point of (3.3) is an extremum point of the function V (.). This fact together with the convergence implies that any trajectory of (3.3) ends at one of the extrema of V (.).

Theorem 3.4 (Hirch & Smale, 1974) Let $x*$ be a strict local minimum of V (.). Then $x*$ is an asymptotically stable equilibrium of the gradient system given in (3.3).

Corollary 3.2:

x^* is an equilibrium point of (3.3) if x^* is an extremum of V(.).

Proof: Necessary conditions for x^* to be an extremum of V (.) is $\nabla V(x^*) = 0$

So, x^* is an equilibrium point of (3.3)

If it is possible to formulate the cost function such that its minima coincides with the stable equilibrium points of a convergent dynamical system, the dynamical system will solve the optimization problem since its steady state solutions will be the minimum points of optimization problem.

3.2.2 Hopfield Networks

One of the well-known dynamic systems used for optimization problems are Hopfield networks. The original Hopfield NNs introduced by Hopfield (1982) consist of a fully connected network of neurons capable of performing computational tasks. Using binary state neurons and a stochastic algorithm to update the neurons, this network served as a content addressable memory that allows for the recall of data based on the degree of similarity between the input pattern and the patterns stored in the memory. This model is known as the discrete and stochastic Hopfield model. In this non-layered recurrent network, the connection weights are assumed to be fixed and symmetric ($w_{ij} = w_{ji}$), and they store information about the stable states of the network. In the case of an excitatory connection, the weights will take positive values; they will be negative in the case of an inhibitory connection or they will be zero in the case of no interaction. Each neuron *i* is described by an internal and an external state. The internal state (net input value) of each neuron is represented by $u_{i,}$ while the external state (output value) by v_i. In this model, the internal states are continuous and the external states are binary. The input of each neuron comes from two sources, external inputs I_i and inputs from other connected neurons. The relationship between the internal and external states of the neurons is represented by the following McCulloch and Pitts dynamics rule.

$$u_i(t+1) = \sum_{j=1}^{N} w_{ij} v_j(t) + I_i \tag{3.5}$$

$$v_i(t+1) = f(u_i) = \begin{cases} 1 & if\, u_i \rangle 0 \\ 0 & if\, u_i \leq 0 \end{cases} \tag{3.6}$$

The internal state of a neuron is found by taking the weighted sum of the external states of all connecting neurons with a constant external input to that neuron. In Eq. (3.5), t is a discrete time, w_{ij} is the synaptic interconnection strength from neuron j to i, f is the activation function between u_i and v_i and can take several forms. It can be the unit step function as defined by the Eq. (3.6).

The states of the neurons are updated in a random manner. The objective function and the problem constraints are mapped onto a quadratic function that represents the energy of the system of neurons.

$$E = \frac{-1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} v_i v_j - \sum_{i=1}^{N} I_i v_i \tag{3.7}$$

Hopfield has proved that with symmetrical weight matrix and non-negative elements on the diagonal of the weight matrix w, the energy function, by performing gradient descent, minimizes until convergence to stable states, which represent the local minimum values of the energy function.

After the original discrete stochastic model based on McCulloch-Pitts neurons was introduced, in a later work, Hopfield (1984) proposed a deterministic model based on continuous neurons. The idea was inspired by the fact that the neurons of the original model were different than the real biological neurons and from the realistic functioning of electronic circuits. So, by maintaining the important properties such as content-addressable memory of the original model, a new model is constructed. The continuous Hopfield model given in Hopfield (1984) is represented by the following resistance-capacitance differential equation to model the capacitance and resistance of a real neuron's cell membrane. In this model, the dynamics of each neuron *i* may be defined as below.

$$\frac{du_i}{dt} = \sum_{j=1}^{n} w_{ij} v_j - \frac{u_i}{\tau} + I_i \tag{3.8}$$

$$v_i = f_i(u_i) = \frac{1}{2}(1 + \mathbf{tanh}(\frac{u_i}{T})) \tag{3.9}$$

$$\tau = RC \tag{3.10}$$

where *t* is a continuous time, *f* is a continuous sigmoidal transfer function that determines the relationship between the internal state of a neuron and its output level, *R* is the trans-membrane resistance, *C* is the input capacitance, *T* is a parameter to control the slope of the transfer function and τ is the value of time constant of the amplifiers. In this model, the external states are ranged between 0 and 1, and are continuous.

3.2.2.1 Stability and Convergence

The energy function for this continuous model has the following form.

$$E = \frac{-1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} v_i v_j - \sum_{i=1}^{n} I_i v_i + \int_{0}^{v_i} f_i^{-1}(v) dv \tag{3.11}$$

Provided that the inverse of f_i exists, it is a Liapunov function and always converges to a stable state. If the weight matrix is symmetric,

$$\begin{aligned}\frac{dE}{dt} &= -\sum_i \frac{dv_i}{dt}\left(\sum_j W_{ij}v_j - \frac{u_i}{R_i} + i_i\right)\\ &= -\sum_i C_i\left(\frac{dv_i}{dt}\right)\left(\frac{du_i}{dt}\right)\\ &= -\sum_i C_i (f_i^{-1})'(v_i)\left(\frac{dv_i}{dt}\right)^2\end{aligned}$$

is the time derivative of E.

$f_i^{-1}(v_i)$ is a monotonically increasing function and C_i is positive, therefore $\frac{dE}{dt} \leq 0$.

$\frac{dE}{dt} = 0$ implies $\frac{dv_i}{dt} = 0 \quad \forall i$, and boundedness of the energy (E) proves that E decreases and converges to a minimum where it stays.

When the slope (gain) of the activation function is very high (T is near zero), the integral term vanishes and we will have the same energy function as in the discrete model. In other words, the stable points of the very high gain, continuous, deterministic model will correspond to the stable points of the original stochastic model.

The idea of using ANNs to provide solutions to NP-hard optimization problems was pioneered by Hopfield & Tank (1985) with the use of their network for solving the Traveling Salesman Problem (TSP). In their paper, Hopfield & Tank (1985) showed that if an optimization problem can be represented by an energy function, then a Hopfield network that corresponds to this energy function can be used to minimize this function and thus provides an optimal or near-optimal solution. Since then, because of the advantages of using Hopfield networks, extensive research has been carried out on the application of the Hopfield networks for solving different optimization problems. Massive parallelism and convenient hardware implementation of the network architecture are among the most important advantages of Hopfield networks.

In this network, objective function and the problem constraints are encoded in terms of an appropriate *energy function*. The aim is to obtain a configuration minimizing the energy function. Translation of the optimization problem into an appropriate energy function is in general, a difficult task. It must be in a quadratic form to meet the form of the energy function of the Hopfield network. Applying the most common method, penalty function approach, the energy function of the network is set equivalent to the objective function of the problem, and the problem is reduced to an unconstrained form by including the constraints of the problem in the energy function as penalty terms. By this way, the constraint violations are penalized. The next step is to compare the energy function of the problem with the energy function given by equation (3.7) to derive the weights and external inputs. Then, by random initialization of the network and updating the neurons, the stable states will be obtained.

However, Hopfield NNs have some shortcomings. They do not guarantee the feasibility. By performing gradient descent on the energy function, the Hopfield model gets easily trapped in local minimum states, and this causes decreasing efficiency especially in large sized problems. Its performance is very sensitive to the initial configuration of the network. Determining the penalty coefficients requires a tedious trial and error process. It requires a large number of iterations to converge to a solution.

In this thesis, we apply Hopfield like neural networks to solve the scheduling problems addressed in this research. The following chapter presents a comprehensive review about neural network applications in production scheduling. The studied problems and the proposed dynamical networks are explained in Chapters 5 and 6 in detail.

CHAPTER FOUR
NEURAL NETWORK APPLICATIONS IN PRODUCTION SCHEDULING

Since the introduction of the first formalized model of a neuron in 1943 by McCulloch and Pitts, there has been a great progress of neurobiology. This progress allowed researchers to build mathematical models of neurons to simulate neural behavior. ANNs can be defined as networks of elementary nodes called artificial neurons or processing elements that are interconnected by direct links called connections and the neurons cooperate to perform parallel distributed processing to solve a desired computational task. Applications of ANNs to production scheduling (for detailed survey see Sabuncuoglu, 1998; Akyol & Bayhan, 2006) are in accordance with using them as a highly parallel model for general-purpose computing and then applying for different combinatorial optimization problems (for detailed survey see Looi, 1992; Smith, 1999). The purpose of this chapter is to give a comprehensive survey of recent research on ANN applications in production scheduling, and to identify some future research directions. We review the scheduling studies with ANNs beginning from the Hopfield network method usage to the probable practices of neural networks evolved with recent developments in evolutionary algorithms.

4.1 ANNs in Scheduling

Artificial Neural Networks (ANNs) can be put into local search based metaheuristics category which includes simulated annealing, noisy methods, guided local search methods, iterated local search, tabu search, threshold accepting, and variable neighborhood search (Osman, 2002). From a modeling viewpoint, they are mathematical representations of biological nervous systems that can carry out complex cognitive and computational tasks. They are composed of many nonlinear interconnected processing elements that are analogous to neurons, and connected via weights that are analogous to synapses.

The modern age of neurocomputing started with the work of McCulloch & Pitts (1943) in which the first mathematical model of a single biological neuron was presented. Although McCulloch and Pitts' study showed that simple type of neural networks were able to learn arithmetic or logical functions, ANN algorithms have been successful enough for many applications in the mid 1980s (Potvin & Smith, 2003). The field attracted the attention of many researchers from different disciplines such as engineering, physics, mathematics, computer science and medicine. In recent years, ANNs have become popular in various real world applications including prediction and forecasting, function approximation, clustering, speech recognition and synthesis, pattern recognition and classification, and many others.

In the scheduling literature, ANNs have attracted much attention because of their characteristics listed below.

- ANNs capture the complex relationship between the input and output variables that are difficult or impossible to analytically relate after they are exposed to examples of the relationship, that is, after they learned. After they learned the unknown correlation between the input and output data, they can generalize to predict or classify for cases they were not exposed to.

- In some cases of designing manufacturing systems, ANNs are preferred to time consuming simulation approaches.

- As a schedule retrieval system, ANNs such as backpropagation networks (BPNs) produce a schedule for a given set of input parameters but unlike the Hopfield networks; they do not generally perform optimization.

- BPNs are also used to select scheduling rules or a manufacturing strategy to achieve accurate estimations of parameters such as the values of the look ahead parameters of scheduling rules. They are used to estimate the system performance measures such as mean utilization, mean job tardiness, mean flow time, etc.

- In static scheduling environments, it is possible to obtain the optimal or near optimal schedules by mathematical modeling, dynamic programming, branch and bound or other advanced methods. But, since real manufacturing environments are dynamic, flexible scheduling methods are needed to react any change in the system that varies with time. So, in dynamic scheduling environments, ANNs are employed to reduce the need for rescheduling.

- While optimizing networks such as Hopfield network and its extensions are involved directly in the optimization by mapping the scheduling objective functions to be optimized and constraints of the problems on to these networks, competitive networks can detect regularities and correlations in input vectors and adapt future responses accordingly (Min, Yih, & Kim, 1998).

In recent years, besides their advantages of parallelism, learning, generalization capability, nonlinearity, and robustness, several limitations of ANNs such as settlement into local minima, trial and error parameter determination process, long learning time are perceived. To compensate their disadvantages, hybrid systems in which ANNs are combined with traditional heuristics or metaheuristics and/or evolutionary algorithms or different approaches, and evolutionary ANNs have been proposed in ANN literature.

4.2 Scheduling with Stand Alone Neural Networks

4.2.1 Hopfield Network and Its Extensions

The original Hopfield NNs, which consist of a fully connected network of neurons capable of performing computational tasks were introduced by Hopfield (1982). Using binary state neurons and a stochastic algorithm to update the neurons, this network served as a content addressable memory that allows for the recall of data based on the degree of similarity between the input pattern and the patterns stored in the memory. This model is known as the discrete and stochastic Hopfield model.

In a later work, Hopfield (1984) proposed a deterministic model based on continuous neurons. The idea was inspired by the fact that the neurons of the original model were different than the real biological neurons and from the realistic functioning of electronic circuits. So by maintaining the important properties such as content-addressable memory of the original model, a new model is constructed.

The idea of using ANNs to provide solutions to NP-hard optimization problems was pioneered by Hopfield & Tank (1985) with the use of their network for solving the Traveling Salesman Problem (TSP). In their paper, Hopfield & Tank showed that if an optimization problem can be represented by an energy function, then a Hopfield network that corresponds to this energy function can be used to minimize this function and thus provides an optimal or near-optimal solution. Since then, because of the advantages of using Hopfield networks, extensive research has been carried out on the application of the Hopfield networks for solving different optimization problems. Massive parallelism and convenient hardware implementation of the network architecture are among the most important advantages of Hopfield networks.

In this network, objective function and the problem constraints are encoded in terms of an appropriate *energy function*. The aim is to obtain a configuration minimizing the energy function. Translation of the optimization problem into an appropriate energy function is in general, a difficult task. It must be in a quadratic form to meet the form of the energy function of the Hopfield network. Applying the most common method, penalty function approach, the energy function of the network is set equivalent to the objective function of the problem, and the problem is reduced to an unconstrained form by including the constraints of the problem in the energy function as penalty terms. By this way, the constraint violations are penalized. The next step is to compare the energy function of the problem with the energy function of the Hopfield network to derive the weights and external inputs. Then, by random initialization of the network and updating the neurons, the stable states will be obtained.

The success in applying neural networks to the TSP motivated many scheduling researchers to employ Hopfield networks. Foo & Takefuji (1988a,b) used a two dimensional Hopfield TSP type matrix of neurons with *mn+1* rows and *mn* columns where *m* and *n* are the number of machines and the number of jobs, respectively, to map their job shop scheduling problem on. To find the global minima of the energy function that represents the objective function of the problem, they applied simulated annealing (SA) which is a stochastic optimization technique and uses a stochastic hill-climbing algorithm with the added ability to escape from local minima in the state-space where conventional methods usually get trapped (Kirkpatrick, Gelatt, & Vecchi, 1983). From the results obtained, it is seen that the proposed methodology gives near optimal solutions rather than an optimum solution. In order to get better results and to reduce the number of neurons required to solve the same problem, Foo & Takefuji (1988c) introduced integer linear programming networks as extensions of the original Hopfield network, and achieved better solutions. But, in his paper, Van Hulle (1991) addressed that the network of Foo & Takefuji (1988c) generated constraint-violating solutions. To overcome this drawback, the original job shop scheduling problem was formulated again as a goal programming problem to be mapped onto a goal programming network. The simulation results showed that although the proposed approach yielded feasible solutions it could not guarantee optimal solutions.

The limitations of the traditional Hopfield NNs based on the quadratic energy function triggered the authors Zhou, Cherkassky, Baldwin, & Olson (1991) to propose a neural network having a linear cost (energy) function rather than the quadratic energy function of the Hopfield network. Doing so, they aimed to improve the scaling properties of the Hopfield NNs. They compared their network with integer linear programming neural network of Foo & Takefuji (1988c) and TSP type Hopfield network method of Foo & Takefuji (1988a,b) in terms of the number of neurons and interconnections required. The results obtained were very encouraging for both criteria.

Due to the problems of Hopfield NNs in solving optimization problems, various modifications are proposed to improve the convergence of the Hopfield network. While several authors modified the energy function of the Hopfield network to improve the convergence to valid solutions (Brandt, Wang, Laub, & Mitra, 1988; Van Den Bout & Miller, 1988; Aiyer, Niranjan, & Fallside, 1990), many others studied the same formulation with different penalty parameters (Hedge, Sweet, & Levy, 1988; Kamgar- Parsi & Kamgar- Parsi, 1992; Lai & Coghill, 1992). But although the modified versions of the Hopfield network could give valid solutions, they may not converge to good quality solutions. In the following years, poor solution quality of Hopfield networks was improved by integrating stochasticity into the Hopfield network. Boltzmann machine, Gaussian machine, Cauchy machine and mean field annealing approaches were obtained by embedding stochastic properties into the Hopfield network.

A stochastic neural network for solving dynamic resource constrained scheduling problems was proposed by Vaithyanathan & Ignizio (1992). The authors represented their problem as a series of multidimensional knapsack problems, and used neural networks to solve these problems. The network included the combination of a Hopfield network and external neurons to give stochastic property. The experimental results showed that the network was able to avoid local minimum. As mentioned before, Gaussian machines developed by Akiyama, Yamashita, Kajiura, & Aiso (1989) as another alternative approach of escaping local minima were proposed for improving the efficiency and speed of the Boltzmann machine. Like continuous Hopfield networks, they have continuous outputs with a deterministic activation function but random noise is added to the external input of each neuron. In 1992, Arizono, Yamamoto, & Ohta proposed a Gaussian machine model for solving the single machine scheduling problem having the objective of total actual flowtime minimization. Computational results showed that in most of the problems the proposed network was successful in finding the optimal solutions.

Lo & Bavarian (1993) extended the gradient approach of two dimensional Hopfield network to a three dimensional matrix, called neural box, in which the third dimension was the time. They used this network to solve the job shop scheduling and multiple traveling salesmen problem. Although the simulation results showed that the presented approach yields feasible schedules, too many numbers of neurons and interconnections are required for solving large sized problems.

Another extension of Hopfield network was proposed by Satake, Morikawa, & Nakamura (1994) for minimizing the makespan of the job shop scheduling problems. In the energy function, only one constraint is included, and the other constraints are reflected in the threshold values. The difference between the proposed network and the original Hopfield network was the revision of the threshold values of the network at each transition of neurons, and the inclusion of the Boltzmann machine (Hinton & Sejnowski, 1986) known as the integration of the dynamics of the discrete Hopfield model with the simulated annealing methodology. The simulation experiments showed that the presented network gives optimal or near optimal solutions. Following this work, Foo, Takefuji, & Szu (1995) proposed a modified Hopfield and Tank network for job shop scheduling problems. The presented network, used for solving integer–linear programming problems, differs from the traditional Hopfield and Tank network with the addition of nonlinear step function *h* amplifiers and with the use of a linear energy function rather than the quadratic energy function of the original Hopfield and Tank network. They examined the proposed approach, and concluded that it requires more number of neurons and interconnections than those needed by the approach in Zhou *et al.* (1991) that includes a linear energy function, but does not need extensive calculations as in Zhou *et al.* (1991).

In another study, Willems & Brandts (1995) mapped the sequencing and resource constraints of the integer linear programming representation of the job shop scheduling problem on an extension of Hopfield network that includes general rules of thumb as an optimization criterion. The comparison of the proposed approach with heuristic rules showed that it produced better solutions than the traditional heuristic approaches.

Besides its advantage of escaping from local minima, the Boltzmann machine requires large computational times as the size of the problem increases (Aarts & Korst, 1989). In order to reduce the excessive computation times of the Boltzmann machines, Peterson & Anderson (1987) proposed mean field annealing by replacing the stochastic bipolar state neurons of the Boltzmann machine with deterministic and continuous neurons. The normalized mean field annealing (MFA) and the Hopfield neural network method (HNN) were applied to the *n* job *m* machines scheduling problem including resource and timing constraints in Huang & Chen (1999). To solve the problem, neural net optimization algorithm was used. In other words, states that both satisfy the constraints of the problem and minimize the energy function were found. In this work, rather than using linear programming or the *k* out of *N* rules to define the energy function, the objective function was formulated according to the constraints involved, step by step. Then the total energy with all constraints was obtained. The derived energy function was transformed into corresponding neural network for both algorithms HNN and MFA. Simulations results showed that the generated energy functions work successfully for multiprocessor problems.

Chen & Dong (1999) studied a production scheduling problem in a major surface mount technology (SMT) factory in Western Canada to minimize the total setup cost in producing different products in one of the SMT assembly lines. A nonlinear mixed integer programming model was proposed to represent the problem with constraint equations. In order to solve the optimization problem, Hopfield-Tank neural network was used. The authors concluded that the computational times to reach optimal solutions using the network approach were comparable to those required by mathematical programming softwares, and believed that significant reduction could be obtained in computational time if parallel computing were utilized.

Liansheng, Gang, & Shuchun (2000) developed an intelligent scheduling model by implementing a unified neural network algorithm. Their network was based on Hopfield neural network, and used to solve different schedule mode problems including job-shop scheduling, priority scheduling, dynamic scheduling, and JIT scheduling.

In a recent work, to deal with the earliness and tardiness multi machine scheduling problem including sequence dependent setup times, Akyol & Bayhan (2005) suggested a coupled gradient network approach which was the extension of Hopfield (1984) and Hopfield & Tank (1985). The aim of their study was to minimize the weighted sum of the earliness and tardiness penalties using a neural network approach rather than the traditional approaches in scheduling. Using the penalty function approach, the formulated problem was represented by an energy function. After six recurrent networks were designed, the dynamics are defined by gradient descent on the energy function. Although, the authors explained the necessary steps to simulate their networks, to test the network was left to a further study.

Any optimization problem of scheduling that can be defined by a quadratic form, can be tackled with Hopfield networks. Then, a Hopfield network whose energy function reaches its minima at the same points with the cost function that describes the scheduling problem must be designed. However, by performing gradient descent on the energy function, the Hopfield model gets easily trapped in local minimum states, and this causes decreasing efficiency especially in large sized problems. Additionally, determining the appropriate values of the penalty parameters, network parameters and initial states are other critical issues associated with this model. Solving scheduling problems represented by many constraints will cause a tradeoff between the penalty terms to be minimized. Despite the promising results obtained by the proposed methods, some aspects still need further studying. There is no exact method that guarantees a global optimum solution. Even if it is achieved, the proposed models will suffer from extremely large computation times. Moreover, few studies are carried out for the comparison of the Hopfield network's and its extensions' performance with best known heuristics' or metaheuristics'. So, we believe this issue will be given more importance in the near future.

4.2.2 Multilayer Perceptrons

One of the important types of networks used in scheduling applications is a multilayer perceptron, a feedforward network including a set of neurons connected by weighted links. It consists of an input layer, one or more hidden layers and an output layer. Backpropagation, which was first introduced by Werbos (1974) and was later rediscovered independently by Parker (1985), and Rumelhart, Hinton, & Williams (1986), and then modified in various manners by numerous researchers in order to overcome its deficiencies, is one of the most popular algorithms for training multilayer perceptrons. This learning rule is a kind of gradient descent technique with backward error propagation, used to adjust the neural weights of a multilayer perceptron. Multilayered perceptrons trained with backpropagation learning algorithm are generally referred to as backpropagation networks. The weights of the network are randomly initialized before training starts. Then, a pair of patterns including the input patterns and the desired patterns is applied to the network. By propagating through the network layer by layer, a set of outputs is produced as the actual outputs of the network. At the output layer, the actual outputs are compared to the desired outputs, and an error signal is computed by subtracting the actual value from the desired value. This error signal is propagated backward through the network and the weight values are then adjusted by a magnitude proportional to the negative gradient of the error function, which is generally equal to the sum of squared errors. By this way, the difference between the actual and the desired outputs is minimized (Haykin, 1994).

Backpropagation networks have been successfully used in modeling, classification, forecasting, design, control, and pattern recognition. Their improved generalization capabilities over competing machine learning tools and their easy mechanism made them attractive to be utilized in production scheduling. A successful use of a backpropagation network for job shop scheduling environments can be found in Chryssolouris, Lee, & Domroese (1991) where they employed simulation and a backpropagation network to establish adequate weights between the operational policy of a work centre and performance measures such as the mean costs, mean flow time and mean tardiness. This study is among the first examples to the neural network based metamodels for the system design problems. A similar application can be found in Philipoom, Rees, & Wiegmann (1994), where a backpropagation network was proposed to determine due dates for job shops. In order to see whether neural networks were successful in assigning due dates, the assigned due dates were compared to regression based due date assignment rules. The results of this study indicated that the neural network gives better results than the six linear rules and the nonlinear regression model with respect to mean absolute deviation and standard deviation of lateness criteria.

Because of their flexibility and adaptability properties, ANNs have been used not only in static scheduling environments but also in dynamically changing manufacturing environments where the values of the system attributes change continually (Chen & Muraki, 1997; Chen, Huang, & Centeno, 1999; Arzi & Iaroslavitz, 1999; Li, Chen, & Lin, 2003).

Geneste & Grabot (1997) showed how to consider the information based on the workshop and the manufacturing orders structure and on the objectives of the workshop manager in order to select a relevant scheduling strategy. They proposed parameterized scheduling heuristics and suggested two methods to tune the heuristic rule.

As pointed out by Jain & Meeran (1998) some of the main problems faced in the application of traditional backpropagation networks and in Hopfield networks are the lack of generalized learning capability to map inputs and outputs for NP hard problems, and the growing network size for large size problems, respectively. To overcome these shortcomings, the authors suggested a modified backpropagation model and used it for makespan minimization. The main difference between the proposed network and other backpropagation networks was that it performed optimization itself. The modified backpropagation system was compared with three priority dispatching rules; SPT, MWR, FCFS, and the Shifting Bottleneck Procedure of Adams, Balas, & Zawack (1988). The proposed system offered shorter makespans in considerable computational times than three dispatching rules.

For dealing with single machine sequencing problems, El-Bouri, Balakrishnan, & Popplewell (2000) developed a backpropagation neural network approach where they utilized a 11-9-1 three layered neural network in which each job is represented by its specific information and the output unit determines where the corresponding job lies in the sequence. The proposed network is evaluated for three performance criteria; mean flow time, mean weighted flow time, and maximum job tardiness. The network is successful in minimization of the mean flow time and the mean weighted flow time. The network also allows the jobs to be sequenced in order to minimize the maximum tardiness. For another performance criterion, minimization of the mean job tardiness, the network's capability was investigated and the results were compared with two sorting rules. Although the network's solutions are superior those of the sorting rules, about 6-12 % difference from optima motivated the authors to develop a Neural Job Classification and Sequencing System (NJCASS). The results showed that NJCASS has many advantages, for instance, was flexible under different performance criteria. The approach proposed by Hamad, Sanugi, & Salleh (2003) bears some similarities to that of El Bouri *et al.* (2000), although the former is applied to a single machine case. Hamad *et al.* (2003) dealt with the non-identical parallel machines problem and proposed a way of representing the problem to be fed into a backpropagation network, and tried to minimize the sum of earliness and tardiness costs. In this study, the two-output representation is used instead of one-output unit (representing the target values) representation proposed in El-Bouri *et al.* (2000).

In their work, Park, Kim, & Lee (2000) presented a neural network approach for solving identical parallel machine scheduling problems with sequence dependent set up times to minimize weighted tardiness. Their work is an extension of Kim, Lee, & Agnihotri (1995)'s approach to parallel machine situation. The difference between them is the inclusion of an additional factor called set up time range factor. The presented approach is also an extension of Lee, Bhaskaran, & Pinedo (1997)'s ATCS

(Apparent Tardiness Cost with Setups) rule in which four factors are used to quantify the problem characteristics. The differences between them are that the proposed approach includes an additional factor, and also trains a backpropagation network to obtain the values of the look ahead parameters. The simulation experiments show that proposed extended approach causes 4 % improvement over the original ATCS rule.

Sabuncuoglu & Touhami (2002) used backpropagation networks as a simulation metamodel, and tried to measure metamodel accuracy in estimating manufacturing system performances in the job shop scheduling environments. The numerical results showed that metamodelling with neural networks can be used effectively to estimate the system performances. Another neural network based metamodel application can be found in the study of Fonseca & Navaresse (2002) that shows the use of ANNs as a valid alternative to the traditional job shop simulation approach. In order to generate the training and test sets, the simulation software package Arena was used and applied to a problem from Askin & Standridge (1993). From the simulation analysis, average flowtimes were estimated when job types followed different machine sequences. It is seen that the average flowtimes obtained from three different simulation packages, i.e. Arena, SIMAN, and ProModel were almost identical to the simulation outputs of the developed neural network models.

In another study, Raaymakers & Weijters (2003) also used backpropagation networks to estimate the makespan of job sets in batch process industries. Because the amount of job interaction depends on the mix of the jobs and the resource sets, they use aggregate characteristics of the jobs and the resources to estimate the amount of interaction. The authors applied both neural networks and regression analysis to determine the relationship between the variables affecting the amount of interaction and the amount of interaction at the scheduling level. Two kinds of regression models were used in this study; the first one included only main effects, and the other comprised main effects and also two way interactions. The computational results showed that these regression models and neural networks give satisfactory solutions, but the neural network’s estimation quality is significantly better than these models.

Cha & Jung (2003) addressed the schedule assessment problems with the complex and competing environment of manufacturing systems. In order to overcome this problem, they introduced a methodology to provide a consistent and dimensionless degree of satisfaction. They exploited fuzzy numbers to represent the final assessment result of a schedule.

Feng, Li, Cen, & Huang (2003) applied multilayered perceptron networks to design, develop and implement a production activity scheduling system to be used in a job shop environment. They presented a different data encoding method to represent the processing time and processing sequence of the jobs to be processed, used backpropagation training algorithm to control local minimum solutions, and introduced a heuristic method for revising the initial output. The implementation of the developed scheduling system on a real life job shop problem helped to improve the production measures of the manufacturing plant.

Cakar & Cil (2004) applied backpropagation networks for the design of manufacturing systems. Performance measures such as mean flow time, mean tardiness, maximum completion time, machine utilization rate of each work center and percentage of late parts are fed as inputs into the neural network, and the number of machines in each work center is obtained as output from the system. For each of the priority rules, namely, earliest due date, shortest processing time, first come first served and critical ratio rules, the shop is simulated and four different training sets are formed. For four priority rules, four networks are trained by using these training sets. Four different design alternatives are obtained and these design alternatives are evaluated according to the performance measures, and the best alternative is chosen.

In addition to the above studies, Akyol (2004) exploited backpropagation networks to model six different heuristic scheduling algorithms applied to a makespan minimization problem of a flowshop. They incorporated fuzzy representation into their preprocessing steps and then, trained their networks. The results obtained from the comparison of proposed approach with the six heuristic algorithms showed that the proposed method was successful to predict the makespan of the *n* job *m* machine permutation flowshop environment.

When the articles reviewed above are considered, it can be said that backpropagation networks, except the study of Jain & Meeran (1998), are not directly involved in the optimization problem. That is, actual scheduling is not performed. The success of most of the studies are the result of the good generalization capabilities of backpropagation networks which are used to capture the complex relationship between the input and output variables of the considered scheduling problem. Additionally, as also pointed out by Sabuncuglu & Tohami (2002), in recent years, for the design of manufacturing systems, the literature includes different neural network based metamodels in which the training data is provided by simulation. Despite the increase in the training time, integration of simulation with neural networks will provide better results in less time compared to time consuming stand alone simulation approach. Although the popularity of backpropagation networks has grown significantly in the past few years, some problems still exist with the application of the backpropagation networks. That is, these networks are trained by a gradient based search technique which has the risk of getting stuck in local optimum and the starting point of the connection weights becomes an important issue to reduce the possibility of being trapped in local optimum. Another difficulty with the construction of these types of networks is the necessity of generating a training set which is time consuming. Therefore, in recent years, the performance of these networks is tried to be enhanced by combining them with different heuristics or metaheuristics.

4.2.3 Competitive Networks

The works by Grossberg (1972), von der Malsburg (1973), Fukushima (1975), Willshaw & von der Malsburg (1976), and Grossberg (1976a,b) are the first in the area of competitive learning. Unlike Hopfield networks, *the winner take all* strategy forms the base of the competitive networks. In this unsupervised network, there is a single layer of output neurons fully connected to the input neurons of the network. In this output layer known as the competitive layer, lateral inhibition occurs among the neurons, and each neuron tries to inhibit the neuron to which it is laterally connected. For an input pattern presented to the network, the neuron with the weight vector at the least distance from the input vector is called the winner and its output is set to one.

In order to apply competitive networks to solve optimization problems, the equations of motion for the problem constraints and an energy function that converges to stable states must be defined. For detailed information one can refer to Fang & Li (1990). Fang & Li (1990) obtained equations of motion for the 0-1 knapsack problem, the generalized assignment problem and the single machine total tardiness scheduling problem including unit processing times and different deadlines. Although their study generated good results, the literature on the application of competitive networks to scheduling is sparse. More work has to be done in deriving the equations of motion to represent different constraints present in different types of scheduling problems.

A neural network model including a three dimensional structure as in the work of Lo & Bavarian (1993) was proposed by Sabuncuoglu & Gurgun (1996). It is very similar to the Hopfield network but includes an external processor for monitoring and controlling the network evolution. The difference between the Hopfield network and the proposed network is that the proposed network involves a competition property. In other words, the neurons (jobs) compete with each other to be in the first available position in the sequence. This network was employed for solving the single machine mean tardiness problem, and job shop scheduling with makespan minimization. The performance of the proposed network was compared with the Wilkerson and Irwin (WI) algorithm, in terms of mean tardiness and the computation time, and gave better solutions than WI. Then, the performance of the proposed approach for job shop scheduling was tested on a number of job shop scheduling problems, and the proposed network found optimal solutions in most of the problems.

Chen & Huang (2001) applied a competitive neural network in order to obtain solutions to the multiprocessor job scheduling problem with multiprocesses. The problem involves time and resource constraints, and is depicted by an energy function proved to be converging. This function is mapped onto the competitive Hopfield neural network (CHNN) known as a Hopfield neural network (HNN) with a *winner-take-all learning* mechanism. In other words, in competitive Hopfield neural network, instead of conventional deterministic learning rules, a competitive learning mechanism is used to update the neuron states so that the time required in obtaining coefficients is reduced and effective results are obtained. The simulation experiments showed that the method was successful.

Based on competitive learning, Kohonen (1982) proposed an unsupervised, clustering network known as self-organizing map in which only one neuron per group is on at a time. McMullen (2001) developed a neural network approach of the Kohonen self-organizing map (SOM) for solving a JIT production-sequencing problem with setups minimization and material usage stability. The experimental results based on various test problems from the literature shows that SOM approach gives near optimal solutions with respect to the objectives considered, and its overall performance is competitive with the search heuristics such as simulated annealing, tabu search and genetic algorithms (GAs). But the proposed method needs more efforts to handle the CPU time problem.

In their later work, Min & Yih (2003) integrated simulation and a competitive neural network trained with the Kohonen learning rule, and developed a multi-objective scheduler to select dispatching rules for both machine and vehicle initiated dispatching decision variables, and to obtain the desired performance measures at the end of short production intervals. Extensive simulation experiments were conducted to collect the data including the relationships among the change of decision rule set and current system status and the performance measures of a semiconductor wafer fabrication system. A competitive network was used to group all instances of simulation outputs.

4.3 Scheduling with Hybrid Approaches

Several shortcomings of ANNs motivated the researchers to integrate neural networks with different computing techniques. As a result, to enhance the performance of the neural networks, there has been an explosive growth in the successful use of hybrid neural networks in scheduling. In this section, we review the scheduling studies exploiting the combinations of neural networks with different approaches.

Rabelo & Alptekin (1990) introduced a hybrid approach using expert systems and backpropagation neural networks, and applied their hybrid system to find solutions for the FMS scheduling/rescheduling problem. To choose the best scheduling rules with respect to different criteria, ANNs were used to identify patterns in the tasks to be solved, and expert systems were used to monitor the performance of the system and to automate the learning process of the ANN.

ANNs are combined with GAs, first proposed and studied by Holland (1975), to handle *trapping in local minima*, one of the important shortcomings of ANNs. Works in the area of GAs were summarized in Schaffer, Whitley, & Eshelman (1992).

Dagli & Sittisathanchai (1993) also proposed a hybrid approach combining GAs with neural networks. Firstly, they tested their approach using a problem from Foo & Takefuji (1988), and showed that the hybrid method found the optimum solution in a few iterations. Then, even the number of machines and jobs were increased, the results were also encouraging. Furthermore, the genetic neuro-scheduler proposed by these authors produced better solutions than the shortest processing time (SPT) rule for different sizes of problems. Another GA including hybrid approach system for selecting candidate scheduling rules, which minimize the maximum tardiness and mean flow time, from a larger list of rules was developed by Rabelo, Yih, Jones, &

Tsai (1993) where backpropagation neural networks, parallel Monte Carlo simulation and inductive machine learning mechanism were integrated. The test results proved the success of the approach.

In recent years, the development of artificial intelligence techniques has provided a powerful way of dealing with dynamic scheduling problems. In the study of Sim, Yeo, & Lee (1994), the backpropagation neural network is integrated with an expert system for solving dynamic job shop scheduling problems, and by this way, the weakness of each stand alone method is tried to be overcome. The integrated method exploits the advantages of both techniques, that is, the expert system helps to reduce the training time of the neural network by training sub-networks separately, while the neural network learns about and handles the complex interactions of the scheduling considerations without the need for the long knowledge acquisition and development time of expert systems. The authors showed that the proposed network has better performance than priority dispatching rules, and could tackle with the adaptive scheduling problems.

One of the major drawbacks encountered with neural networks is their lack of explanation power. It is difficult to explain how the networks arrive at their solutions due to the complex non-linear mapping of the input data by the networks. In many applications, to gain better understanding of the problems at hand it is desirable to induce knowledge from trained neural networks. In the literature, applying machine learning techniques to extract dynamic scheduling knowledge has been a successful method. In their work, Li, Wu, & Torng (1997) combined an adaptive neural network classifier and a decision tree technique to obtain scheduling knowledge for flexible manufacturing systems. System performance data are fed into the adaptive resonance theory neural network model (Carpenter & Grossberg, 1987) as inputs, and classified according to the similarities between them. In order to find a definition for each class, a decision tree method is performed and then this is converted into a set of rules to be used as the real time scheduling knowledge.

In the same year, in order to overcome the problems of convergence, stability and sensitivity to the initial inputs belonging to Hopfield networks, Jeng & Chang (1997) presented a non-energy based neural network architecture that implements a heuristic rule, combination of most-valid operation first and shortest operation first rule. They used this network to solve job shop scheduling problems with makespan minimization, and obtained optimal or near optimal schedules.

Lee & Dagli (1997) designed a parallel genetic-neuro scheduler including six different modules, for solving large size job shop scheduling problems, and tested it on different size of job shop scheduling problems. The results show that the developed approach is able to reach to the optimum solution in a few iterations, and has superior solutions to SPT, EDD, SLACK for minimizing the lead time.

Min, Yih, & Kim (1998) designed a dynamic and real time FMS scheduler by combining the competitive neural network and search algorithm to meet the multiple objectives given by the FMS operator. Based on the current decision rules, current system status and performance measures, the competitive network generates the next decision rules. The simulation results indicate that the FMS scheduler is able to satisfy multiple objectives given by the operator. Another multiple objective flexible manufacturing system (FMS) scheduler was developed by Kim, Min, & Yih (1998) with the same objective. Their approach is the integration of inductive learning, competitive neural network and simulation. They compare, for different objectives, the competitive network approach with the proposed integrated approach. The results show that the use of inductive learning is effective to refine the rough scheduling knowledge.

Rather than the usual non-adaptive neural networks proposed in the literature, Yang & Wang (2000) proposed a constraint adaptive neural network (CSANN) for the generalized job shop scheduling problem that is more complex than the traditional job shop scheduling problem. The problem is represented by the integer mathematical programming models, and then mapped onto a neural network that consists of two layers. In this study, three different heuristic algorithms are combined with the proposed CSANN. From the simulation experiments conducted, it is seen that the performance of CSANN is improved by combining CSANN with the proposed heuristics. Later, Yang & Wang (2001) extended the work of Yang & Wang (2000). In the latter approach, a new heuristic based on obtaining a non-delay schedule, and one of the heuristics in Yang & Wang (2000) used to increase the speed of the solving process of CSANN, are combined with CSANN to form a new hybrid approach for job shop scheduling problems. According to the simulation experiments, the new hybrid approach is efficient in obtaining the minimum makespan, and is fast in making calculations. Another constraint neural network was introduced by Yu & Liang (2001) where they again try to solve the expanded job shop scheduling problem (EJSSP), which is more difficult to solve than the original job shop scheduling problem, by involving additional constraints such as job delivery due dates and available time of the resources. They propose a hybrid approach of neural networks and GAs. In order to describe the processing constraints and resolve the conflicts, three types of neurons are described. Then a constraint neural network (CNN) formed by these neurons is developed. To optimize the starting time of the EJSSP, a gradient CNN is constructed. This gradient CNN is combined with GA for optimizing the sequence of the scheduling problem. The results of the study show that the hybrid approach is effective for complex scheduling problems.

To deal with fuzzy and random production disturbances faced commonly in manufacturing systems, Li, Li, Li, & Hu (2000) presented a production rescheduling expert simulation system based on Chinese manufacturing. It combines many different techniques and methods, including simulation, backpropagation neural network, expert knowledge and dispatching rules. The simulation module provides training patterns for the network. Simulation results showed that the production rescheduling expert system is practical and increases production efficiency.

Another use of GA- neural network combination can be found in Lee & Shaw (2000) where they proposed a two level neural network for a real time flow shop sequencing problem of a printed circuit board (PCB) manufacturing environment.

Firstly, they compare the performance of their pure neural network with two constructive heuristics: the deterministic greedy search and the NEH heuristic (Nawaz, Enscore, & Ham, 1983). For this reason, they construct a total of 10 problem sets including different number of machines and different number of jobs. Simulation results from 30 runs for each problem set are averaged. The comparison of the makespan and computational times by the neural network approach and constructive heuristics indicate that the neural network approach is superior. Then, the neural network approach's performance is also compared with GAs. The results show that the neural network's performance are within 3.4% of those of GAs but the computational time needed by the neural network is only less than 0.2 % of that of GAs. The last implementation of this work is the combination of the neural network approach with GAs, and the simulation results indicate that the combined algorithm improves the solution quality and computational time of the GAs.

From optimization viewpoint, the Hopfield neural network and its extensions belong to the penalty method for solving the constrained real optimization into which a combinatorial optimization is converted. The penalty function requires the weighting factors for the penalty terms to be sufficiently large in order to converge to a feasible solution. But as the penalty terms become stronger, the original objective function becomes weaker, and as they become larger and larger, the problem becomes ill conditioned. To deal with this problem, Li (1996) combined the augmented Lagrange multiplier method and the penalty methods of the Hopfield networks to obtain the augmented Lagrange Hopfield network. By this way, both the solution quality and the convergence properties of the Hopfield network are improved. Thus, the proposed approach helps to overcome the problems associated with the penalty method or the Lagrange multiplier method when used alone (Li, 1996). Following this work, Luh, Zhao, & Wang (2000) proved the convergence of Lagrangian Relaxation Neural Networks (LRNN) for separable convex problems, and constructed LRNN for separable integer programming problems. They applied LRNN to separable job shop scheduling problems. By using Lagrange multipliers, the machine capacity constraints are relaxed, and the relaxed problem is decomposed into sub problems each of which is solved by dynamic programming. The results indicate that the performance of the method is much better than those of the existing neural network approaches in scheduling.

In another study, Liebowitz, Rodens, Zeide, & Suen (2000) incorporated a Hopfield neural network approach into the generically used expert scheduling system (GUESS). GUESS is an intelligent scheduling toolkit developed by Liebowitz, Krishnamurthy, Rodens, Houston, Liebowitz, & Zeide (1997) including a heuristic based approach, a hill-climbing algorithm and a GA approach to scheduling. The performance of the neural network is compared with the other approaches used by GUESS. From the results, it is seen that the neural network approach produces good solutions for scheduling problems.

An altogether different approach was presented by Chen & Huang (2001) for solving the multiprocessor scheduling problem involving non-preemptive multitasking with timing constraints. The proposed network known as a fuzzy Hopfield NN (FHNN), was different from the standard Hopfield network in the sense that a fuzzy c-means clustering algorithm is incorporated into it. In this method, each processor (job) is regarded as a data sample and every processor as a cluster. The objective function to be minimized is defined as the Euclidean distance between the data samples and the cluster sample, and the goal is to find the best set of clusters. The simulation results showed that the modified energy function of the network converges rapidly into a minimum value, and the penalty parameter determination problem, a major shortcoming of Hopfield NNs, is overcome.

Another neural network approach to adaptive scheduling can be found in the study of Shiue & Su (2002). In this approach, the aim is to develop a neural network based adaptive scheduling system to identify the important attributes of the system status and generate scheduling knowledge bases for an FMS system. The authors point out that by selecting important system attributes in manufacturing systems, better performance could be achieved in prediction. They develop an attribute selection algorithm based on the weights of backpropagation networks, to measure the importance of system attributes in a neural network based adaptive scheduling

(NNAS) system. Then, they combine their algorithm with the (NNAS) system and obtain an attribute selection neural network based adaptive scheduling (ASNNAS) system. In order to test the efficiency of the (ASNNAS) system, its performance is compared with the (NNAS) system's performance and with some dispatching rules for different criteria. The results show that the ASNNAS system gives better solutions than the NNAS system for all the performance criteria, and requires less computational effort.

Similar to their previous work, Shiue & Su (2003) developed an attribute selection decision tree (ASDT) based adaptive scheduling system by combining backpropagation networks with a decision tree learning (C4.5 algorithm) approach. This approach differs from Shiue & Su (2002)'s approach in using the decision tree learning algorithm in constructing the scheduling system. The authors compare the classical DT-based approach with ASDT-based approach under different performance criteria. The experimental results show that using an attribute selection algorithm improves the generalization ability of knowledge bases, and causes less computational effort. In a similar work, Priore, Fuente, Pino, & Puente (2003) applied backpropagation networks and inductive learning (C4.5 algorithm) to acquire the scheduling knowledge by which the most appropriate dispatching rule in flexible manufacturing systems is determined. To improve the performance of the scheduling systems, they also propose a module used for generating new control attributes.

Wang, Jacob, & Roland (2003) addressed some limitations associated with traditional neural network models. Among these limitations were the requirement of excessive number of neurons, finding unfeasible solutions and the computational effort required for obtaining a solution. They propose a hybrid neural network approach to solve the flexible flow shop scheduling problem, which is a generalization of flow shop and parallel machine scheduling problems, with the objective of makespan minimization. In this study, the authors exploit the structure of optimization problems and heuristic information, and compare their hybrid network with Ding & Kittichartphayak (1994) heuristics with respect to the computational time and solution quality which is measured by percentage of absolute difference between the solution and the lower bound (or the optimal solution) for some small problems. The results of the study show that the proposed hybrid approach outperform all the heuristics on average and succeeds in dealing with the mentioned limitations.

A different application can be found in Agarwal, Pirkul, & Jacob (2003) where an Augmented Neural Network (AugNN) has been proposed for solving the task scheduling problem. The proposed approach is a hybrid of the heuristic and the neural network approaches, and is used to minimize the makespan for scheduling *n* tasks on *m* identical machines. The heuristics used in this study were: Highest Level First, Highest Level with Estimated Time First, Critical Path with most Immediate Successors First, Shortest Path Time, Longest Processing Time and Random. These six heuristics and AugNN (including these six heuristics and two learning rules) are compared based on three criteria- a) reduction in gap between lower bound solution and heuristic solution, b) number of cases with known optimum solutions, c) number of cases where improvement in makespan occurs over heuristic. 570 problems of various sizes, ranging from 10 to 100 tasks, and from 2 to 5 machines are used for testing the performance of the AugNN over the six single pass heuristics. The results indicate that the suggested network outperforms the single pass heuristics with respect to all the three criteria.

Although, the gradient based search techniques such as the back-propagation are currently the most widely used optimization techniques for training neural networks, it has been shown that these gradient techniques are severely limited in their ability to find global solutions. Global search techniques have been identified as a potential solution to this problem. Glover (1986) proposed a meta heuristic approach, tabu search (TS), as a global search technique. Its popularity has grown significantly in the past few years (Sexton, Allidae, Dorsey, & Johnson, 1998). The work done by Solimanpur, Vrat, & Shankar (2004) is a good example to this integration. The authors proposed a neural network based tabu search method for solving the flow shop scheduling problems and the initial permutation obtained from NEH algorithm is tried to be improved. This method is tested on 23 problems proposed by Taillard

(1993) and compared with the BF–TS approach of Ben Daya & Al-Fawzan (1998) in terms of makespan and computational time. The results show that the proposed neuro-tabu search approach is effective over the BF–TS approach in both criteria, and the tabu effect is reduced exponentially.

4.4 Scheduling with Evolutionary Artificial Neural Networks

In recent years, the design of neural networks by evolutionary algorithms has been given great attention by researchers to develop adaptive systems that can change architectures and learning rules according to dynamic environments (Cho & Shimohara, 1998).

ANNs' performance is closely related with their architecture designs. Therefore, obtaining an optimal architecture design has been an important issue in the ANN field. But, since the basic principles governing the processing of information in neural networks is not well understood, optimal architecture design has been a very difficult task depending strongly on human experts having sufficient knowledge about ANNs and the problem to be solved. A trial and error method is used for the manual design that becomes more difficult and unmanageable as ANN complexity increases. Since the selection of the appropriate topology of a network and the best learning algorithm and its parameters are problem dependent, in the literature there have been many attempts to automate the design of ANN architectures. Constructive and destructive approaches are important classes of approaches used. In the former approach, after starting with a small network, neurons or connections are gradually added to the network in order to satisfy the requirements. On the other hand, the latter approach starts with a large network, and then neurons or connections between neurons are pruned to obtain a suitable network. But the neural network structures obtained by these two approaches are constrained to the predefined subsets. Thus, the given task will be performed with a structure from an assumed architectural class rather than an appropriate one (Fang & Xi, 1997).

There has been a growing interest in using evolutionary search algorithms to eliminate the tedious trial and error work of manual design of ANNs. Evolutionary algorithms include evolution strategies (ESs) (Schwefel, 1981; Schwefel, 1995), evolutionary programming (EP) (Fogel, Owens, & Walsh, 1966), GAs (Holland, 1975; Holland, 1992; Jong, 1975; Goldberg, 1989), and a class of population-based stochastic search algorithms based on the ideas and principles of natural evolution. One important characteristic of these algorithms is that individuals in a population compete and exchange information with each other in order to perform certain tasks (Yao, 1999). Similar to ANNs, they have some advantages of robustness and parallelism. But they differ from ANNs in having global search capabilities that make them an applicable and an appealing approach. By maintaining diversity in the population, EAs can tackle large complex problems that generate many local optima. In contrast to gradient-based search algorithms, they do not use the gradient information. They are less likely to fall into local minima, and can be applied to problems for which little prior knowledge is available (Yao, 1997).

The ANNs designed by the evolutionary process are referred to as evolutionary ANNs (EANNs). In other words, they belong to a special class of ANNs in which evolution is another essential form of adaptation in addition to learning. Using two forms of adaptation, EANNs can adapt to a dynamic environment efficiently and effectively (for more detailed information about evolution of ANNs see Yao, 1999).

In EANNs, evolution is employed at different levels to perform several tasks. At the lowest level, evolution can be employed to evolve weight training. In ANNs, weight training is usually formulated as minimization of an error function, such as the mean square error between target and actual outputs averaged over all examples. Connection weights are iteratively adjusted using training algorithms, such as BP and conjugate gradient algorithms based on gradient descent (Alvarez, 2002). Gradient descent based training algorithms have some disadvantages of getting stuck into a local minimum of the error function when the error function is multimodal and/or nondifferentiable. To overcome this drawback, evolution is introduced to find a near optimal set of connection weights without computing the gradient information

(Fogel, Fogel, & Porto, 1990; Porto, Fogel, & Fogel, 1995; Yao, Wei, & He, 1996; Greenwood, 1997; Köppen, Teunis, & Nickolay, 1997; Islam, Akita, Shahjahan, & Murase, 2000; Mandischer, 2002; Cortez, Rocha, & Neves, 2002; Lu, Fan, & Lo, 2003; Ilonen, Kamarainen, & Lampinen, 2003).

At the next higher level, evolution can be employed to evolve the architecture of ANNs that strongly affects the information processing capabilities of ANNs. (Koza & Rice, 1991; Bornholdt & Graudenz, 1992; Tang, Chan, Man, & Kwong, 1995; Mandischer, 1995; Cho & Shimohara, 1998; Sendhoff & Kreutz, 1999; Schmitz & Aldrich, 1999; Kaikhah & Garlick, 2000; Macleod & Maxwell, 2001; Alvarez, 2002; Wicker, Rizki, & Tamburino, 2002; Igel & Kreutz, 2003) This helps to automate the design of ANNs which is a human experience dependent tedious trial and error work.

At the highest level, evolution can be employed to evolve ANN learning rule, which specifies how to adjust weights in weight training. Because the weight training has traditionally been regarded as a learning process, the evolution of learning rules can be considered as a process of learning to learn weights (Yao & Liu, 1998). For different types of architectures of ANNs under consideration, the ANN training algorithm may have different performance. When there is little prior knowledge about the architecture of ANNs, it becomes very difficult to design an optimal learning rule. By adapting a learning rule through evolution it is assumed that ANN's adaptivity will be enhanced in a dynamic environment. By this way, the relationship between learning and evolution will be modeled. Evolution of learning rules differs from the evolution of connection weights and architectures. While the evolution of learning rules works on the dynamic behavior of ANNs, evolution of connection weights and architectures only deal with static objects in an ANN, i.e. weights and architectures. Research dealing with the evolution of learning rules is still in its early stages. Various studies have been proposed on the evolution of learning rules (Chalmers, 1990; Fontanari & Meir, 1991; Baxter, 1992; Merelo, Pat´on, Ca˜nas, Prieto, & Mor´an, 1993; Crosher, 1993; Kim, Jung, Kim, & Park, 1996; Patel, 1996; Moriarty & Mikkulainen, 1996; Kim, Ahn, & Kang, 2000).

Two major problems with the evolution of architectures without considering weights are noisy fitness evaluation and the permutation problem. In addition to the evolutionary procedures explained above, simultaneous evolution of ANN architectures and connection weights generally produces better results and lessens the impact of fitness evaluation and permutation problem (Fang & Xi, 1997; Pujol & Poli, 1998; Yao & Liu, 1998; Liu & Yao, 2001; Gao, 2003).

Although the researchers deal with combining GAs, a branch of EAs, with ANNs, to the best of our knowledge there has not been any scheduling application including the integration of other evolutionary algorithms with neural networks. It is doubtless that ANN researchers will benefit from the advantages of EAs by complementing and compensating each other's strengths and weakness to tackle the problems of scheduling.

The evolutionary training method can deal with the global search problem of ANNs without computing the gradient information. It will be useful to employ them in solving production scheduling problems for which ANNs are incapable of finding a global minimum. Their application is not restricted to overcome the disadvantages of the backpropagation learning algorithm. EAs can also be used for optimizing recurrent neural networks such as Hopfield networks that possess the weakness of proving a local optimal solution to combinatorial optimization problems including scheduling. The applicability of the same evolutionary algorithm to train different types of networks reduces the human effort needed in developing different training algorithms. Besides having many advantages, EAs are not good at local fine-tuned search. In order to overcome this drawback, they are combined with local search algorithms such as simulated annealing, tabu search, backpropagation algorithm, etc. This kind of hybridization can improve the performance of EAs (Yao, 1991; Mühlenbein, Schomisch, & Born, 1991; Kido, Takagi, & Nakanishi, 1994).

4.5. Summary and Future Research

Over the last decade, ANNs have been applied to an increasing number of real-world problems of considerable complexity and to the theoretical test problems. In this chapter, we tried to provide an extensive literature review on the applications of ANNs to different production scheduling problems. In order to see the gradual development in these works, the recent research studies are summarized in a chronological order. Our survey is limited with the publications appearing in refereed journals and conference proceedings between 1987 and 2005. Table 4.1 summarizes the scheduling applications considered in this paper.

Table 4.1 Evolution of ANNs in scheduling

Year	*Author(s)*	*Approach*	*Application area*
1988 a,b,c	Foo & Takefuji	Hopfield networks	Job-shop scheduling problem
1990	Rabelo & Alptekin	Hybrid of expert systems and backpropagation neural networks	FMS scheduling/rescheduling problem
1990	Fang & Li	Competitive networks	Single machine scheduling
1991	Zhou *et al.*	An extension of Hopfield networks (has a linear energy function)	Job-shop scheduling problem
1991	Chryssolouris *et al.*	Multi layer perceptrons	Job-shop scheduling problem
1991	Van Hulle	Hopfield networks	Job-shop scheduling problem
1992	Vaithyanathan & Ignizio	Hopfield networks	Dynamic resource constrained scheduling problem
1992	Arizono *et al.*	An extension of Hopfield network (A Gaussian machine model)	Single machine scheduling
1993	Lo & Bavarian	An extension of Hopfield network	Job-shop scheduling problem and multiple travelling salesmen problem

1993	Dagli & Sittisathanchai	a hybrid approach combining GAs and neural networks	Job-shop scheduling problems
1993	Rabelo *et al.*	(a hybrid approach) Integration of backpropagation neural networks, parallel Monte Carlo simulation and inductive machine learning mechanism	Job-shop scheduling problem
1994	Satake *et al.*	An extension of Hopfield network	Job-shop scheduling problem
1994	Philipoom *et al.*	backpropagation networks	Job-shop scheduling problem
1994	Sim *et al.*	(a hybrid approach) backpropagation networks integrated with an expert system	Job-shop scheduling problem
1995	Foo *et al.*	An extension of Hopfield network	Job-shop scheduling problem
1995	Willems & Brandts	An extension of Hopfield network	Job-shop scheduling problem
1995	Kim *et al.*	A hybrid approach using heuristic rules and ANNs	Single machine scheduling
1996	Sabuncuoglu & Gurgun	Competitive networks	Single machine and job shop scheduling problems
1997	Chen & Muraki	backpropagation networks	Online scheduling in batch process management
1997	Geneste & Grabot	backpropagation networks	Job-shop scheduling problem
1997	Li *et al.*	(a hybrid approach) Adaptive resonance theory neural network combined with a decision tree technique	FMS scheduling
1997	Jeng & Chang	(a hybrid approach) Non energy based neural network	Job-shop scheduling problems

		(that implemented a heuristic rule)	
1997	Lee & Dagli	(a hybrid approach) Artificial neural network (type is not reported) + GAs	Job-shop scheduling problems
1998	Jain & Meeran	Modified backpropagation network	Job-shop scheduling problem
1998	Min *et al.*	(a hybrid approach) Competitive networks + search algorithm	FMS scheduling
1998	Kim *et al.*	(a hybrid approach) Inductive learning + competitive network + simulation	FMS scheduling
1999	Chen *et al*.	backpropagation network	Scheduling of material handling system (FMS scheduling)
1999	Huang & Chen	Hopfield network and the normalized mean field annealing method (obtained by embedding simulated annealing into the Hopfield network)	Job-shop scheduling problem
1999	Chen & Dong.	Hopfield network	Job-shop scheduling problem
1999	Arzi & Iaroslavitz	backpropagation network	ANN based Production Control System for a Flexible Manufacturing Cell is presented for choosing the most appropriate scheduling rule out of several predetermined ones.
2000	El-Bouri *et al.*	backpropagation network	Single machine scheduling
2000	Liansheng *et al.*	An extension of Hopfield network	Different schedule mode problems including job-shop scheduling, priority scheduling, dynamic scheduling and JIT scheduling.

2000	Park *et al.*	backpropagation network	Parallel machine scheduling problems with sequence dependent set up times
2000	Yang & Wang	(a hybrid approach) Constraint adaptive neural network (CSANN) combined with three different heuristic algorithms	Generalized job shop scheduling problem
2000	Lee & Shaw	(a hybrid approach) Combination of the neural network approach with GAs	Flow shop scheduling
2000	Li *et al.*	backpropagation network	Production rescheduling problems
2000	Liebowitz *et al.*	(a hybrid approach) Hopfield network incorporated into the generically used expert scheduling system	Different scheduling problems
2000	Luh *et al.*	(a hybrid approach) Combination of lagrangian relaxation with Hopfield N.	Job-shop scheduling problem
2001	Chen & Huang	(a hybrid approach) Combination of Hopfield network with fuzzy c-means clustering algorithm	Multiprocessor scheduling problem
2001	Chen & Huang	Competitive networks	Job-shop scheduling problem
2001	McMullen	Kohonen self-organizing map	JIT production scheduling problem
2001	Yang & Wang	(a hybrid approach) Constraint adaptive neural network (CSANN) combined with two different heuristic algorithms	Job-shop scheduling problems
2001	Yu & Liang	(a hybrid approach) A hybrid approach of constraint NNs and GAs	Expanded job-shop scheduling problem

		algorithms	
2002	Li & Ye	Hopfield network	Flow shop scheduling problem
2002	Sabuncuoglu & Touhami	backpropagation network	Job-shop scheduling problem
2002	Fonseca & Navaresse	backpropagation network	Job-shop scheduling problem
2002	Shiue & Su	(a hybrid approach) Neural network based adaptive scheduling system	Flexible Manufacturing Systems
2003	Raaymakers & Weijters	backpropagation network	Batch process industries
2003	Feng *et al.*	backpropagation network	Job-shop scheduling problem
2003	Hamad *et al.*	backpropagation network	Non-identical parallel machine scheduling problems
2003	Cha & Jung	backpropagation network	Job-shop scheduling problem
2003	Li *et al.*	backpropagation network	Flexible Manufacturing Systems
2003	Min & Yih	Competitive networks	Semiconductor wafer fabrication system
2003	Shiue & Su	(a hybrid approach) Backpropagation networks combined with a decision tree learning approach	Flexible Manufacturing Systems
2003	Wang *et al.*	(a hybrid approach) Artificial neural network combined with the structure of the optimization problem	Flexible Flow shop scheduling problem
2003	Priore *et al.*	(a hybrid approach) backpropagation networks and inductive learning	Flexible Manufacturing Systems
2003	Agarwal *et al.*	(a hybrid approach) Neural networks combined with heuristic approaches	Task scheduling problem
2004	Cakar & Cil	backpropagation network	Design of manufacturing systems using different priority rules

		(a hybrid approach)	
2004	Solimanpur *et al.*	A neural network based tabu search method	Flow shop scheduling
2004	Akyol	backpropagation network	Flow shop scheduling problem
2005	Akyol & Bayhan	An extension of Hopfield network	Parallel machine scheduling problem

The conclusions drawn from this detailed review are summarized below.

- Most of the approaches proposed in the reviewed articles are based on Hopfield networks and backpropagation networks, and a great emphasis has been given on the job shop scheduling problem, one of the hardest combinatorial optimization problems encountered in real scheduling environments. The literature presents many variants of traditional ANN approaches to improve their performance by trying to escape from the local minima, by reducing the computational effort required, by speeding convergence and by decreasing the number of neurons and interconnections.

- Although widely preferred in the literature because of their highly parallel computational capabilities, one of the major problems in the application of Hopfield networks to optimization problems is the penalty parameter determination. Due to many constraints needed to express scheduling problems, the energy function will include too many penalty terms that result too many local minima. To satisfy all of the constraints while minimizing the objective function is very difficult and a tradeoff exists between the constraint penalty terms and the objective function term. Thus, we believe that an important direction of future research is to search for the methods to overcome this tradeoff problem. In this regard, rather than using constant penalty parameters during simulations, employing time varying penalty parameters may be offered as a potential solution to this problem. In this thesis, we also used time varying penalty coefficients to solve the problems considered.

- In the last years, ANNs have either been combined with artificial intelligence techniques such as expert systems, with metaheuristics such as GAs, tabu search, simulated annealing or with some heuristic procedures to form hybrid approaches providing superior solutions. As a global search technique, the combination of GAs with ANNs is widely used in obtaining optimal solutions, and considerable success is achieved by overcoming the slow convergence property of GAs and the local minima problem of ANNs. Future research should continue this trend by extending these works.

- In recent years, following the need to solve real world dynamic scheduling problems, rather than non-adaptive neural networks whose connection weights and biases must be prescribed before the networks start to work, adaptive neural networks are developed and their performance is improved by combining them with several heuristic algorithms.

- In the neural network design, setting of the parameters, initialization of the weights, configuration of the network are often problem specific and the correct value of these parameters however is not known a priori. Therefore, for any given problem, a wide variety of parameters must be tried to generate confidence that a best solution has been found. Sensitivity of the ANNs to their initial configuration and inability of the gradient based search techniques to find global solutions motivated the researchers to employ EAs together with ANNs for the automatic adjustment of the parameters and the topology of the ANNs.

- In the dynamic scheduling environments faced in real world manufacturing systems, scheduling and rescheduling problems can be handled by EANN's adaptation and learning properties. While several researchers develop new EAs for ANNS, some try to find remedies for these algorithms' shortcomings such as heavy computational loads, and time-consuming fitness evaluation (Hong, Lee, & Tahk, 2003; Palmes, Hayasaka, & Usui, 2003).

- Together with its advantages, the hybrid approach of EAs and ANNs brings together unsolved problems from two complex areas. While there are many questions that need to be answered, many empirical studies on EANN approaches are being reported for solving different kinds of problems. However, most of the studies have focused on small sized problems, and there are few studies comparing the performance of EANNs with their counterparts. In addition, it is not clearly known at present how performance of EANNs in scheduling is. In order to provide a common platform for comparison, benchmark problems must be generated for different objective functions.

- The review of the EANN literature shows us that evolutionary optimization research area is not fully developed but is growing so fast.

- Not only in the ANN field, the role of local search in the field of EANNs is important. Combining EANNs with local search based metaheuristics which have an important feature of flexibility, will make them more effective and an important alternative to ANNs.

- Last years have witnessed the development of efficient and effective stochastic optimization algorithms such as particle swarm optimization (PSO) algorithm which was first introduced by Kennedy & Eberhart (1995). It is an evolutionary algorithm that simulates the social behavior of organisms such as bird flocking and fish schooling. Due to its easy implementation and effectiveness in performing difficult optimization problems, PSO gained considerable attention among researchers and has been applied to a wide range of problems such as multi-objective optimization problems (Ray & Liew, 2002; Mostaghim & Teich, 2003; Coello Coello & Lechuga, 2002), constrained optimization problems (Hu & Eberhart, 2002; El-Gallad, El-Hawary, & Sallam, 2001; Parsopoulos & Vrahatis, 2002), minimax problems, power and voltage control (Yoshida, Kawata, Fukuyama, & Nakanishi, 1999) and task assignment problems (Salman, Ahmad, & Al-Madani, 2002). Few papers (Tasgetiren, Sevkli, Liang, & Gencyilmaz, 2004; Cagnina, Esquivel, & Gallard, 2004; Allahverdi & Anzi, 2005) report the application of this new emerging algorithm in production scheduling. But due to its advantages over other heuristic methods, it is certain that PSO will be more attractive in the area of scheduling. In the last years, PSO is successfully utilized in evolving ANNs to find optimal weights, appropriate topology and transfer functions. Since standard ANNs need to be customized for each system, integration of ANNs with PSO will make possible to design ANNs automatically, and to obtain good generalization capabilities. Of course, it is unavoidable to see their reflections in the area of scheduling.

We believe that in the near future the researchers will benefit from the use of the recent advances in EAs, ANNs, metaheuristics, and their combinations. It can be concluded that, the future of ANNs not only lies in their explicit use but also lies in its conjunction with other advanced technologies.

CHAPTER FIVE
SOLUTION OF IDENTICAL PARALLEL MACHINE SCHEDULING PROBLEM USING DYNAMICAL GRADIENT NETWORKS

5.1 Introduction

The classical identical parallel machine scheduling problem, given *n* jobs and *m* machines, is to assign each job on one of the identical machines during a fixed processing time so that the schedule that optimizes a certain performance measure can be obtained. Having numerous potential applications in real life, in recent years, various research works have been carried out to deal with the parallel scheduling problems.

The literature of parallel machine scheduling problems has been extensively reviewed by Cheng & Sin (1990) and Mokotoff (2001). Among many criteria, minimizing makespan (maximum completion time) has been one of the most widely studied objectives in the literature. Using the three-field classification introduced in Graham, Lawler, Lenstra, & Rinnooy Kan (1979), the problem is denoted in the scheduling literature as $P||Cmax$ where P designates the identical parallel machines, C_{max} denotes the makespan. We assume, as is usual, that the processing times are positive and that $1<m<n$. The problem is known to be NP-hard in the strong sense (Garey & Johnson, 1979; Sethi, 1977).

Although traditional techniques such as complete enumeration, dynamic programming, integer programming, and branch and bound were used to find the optimal solutions for small and medium sized problems, they do not provide efficient solutions for the problems with large size. Having found no efficient polynomial algorithm to find the optimal solution led many researchers to develop heuristics to obtain near optimal solutions. Though, efficient heuristics can not guarantee optimal solutions, they provide approximate solutions as good as the optimal solutions. These can be broadly classified into constructive heuristics and improvement heuristics.

Most of the algorithms belong to the first category and have known worst case performance ratio (Coffman, Garey, & Johnson, 1978; Friesen & Langston, 1986; Friesen, 1987; Graham, 1969; Hochbaum & Shmoys, 1987; Leung, 1989; Sahni, 1976). The LPT rule of Graham, one of the most popular constructive heuristics, has been shown to perform well for the makespan criterion. This rule arranges jobs in descending order of processing times, such that $p_1 \geq p_2 \geq \ldots \geq p_n$, and then successively assigns jobs to the least loaded machine. The MULTIFIT algorithm, a classical constructive heuristic developed by Coffman *et al.* (1978), determines the smallest machine capacity to find a feasible solution using the LPT scheme. This is achieved by solving heuristically a series of bin packing problems. Although MULTIFIT is not guaranteed to perform better than LPT, it has been shown that it has a worst case bound better than LPT.

Improvement based algorithms are based upon local search in a neighbourhood in which a feasible solution is taken as a starting point and then tried to be improved by iterative changes. Application of these algorithms to the P||Cmax problem can be found in Frangioni, Scutelle, & Necciari (1999), Hübscher & Glover (1994), Jozefowska, Milka, Rozycki, Waligora, & Weglarz (1998).

Although a large number of approaches such as mathematical programming, dispatching rules, expert systems, and neighborhood search to the modeling and solution of scheduling problems have been reported in the literature, over the last decades, there has been an explosion of interest in using Artificial Neural Networks (ANNs) for the solution of various scheduling problems. This is mainly after the success of the use of Hopfield & Tank (1985)'s network in solving the Traveling Salesman Problem. The authors showed that if an optimization problem can be represented by an energy function, then a Hopfield network that corresponds to this energy function can be used to minimize this function to provide an optimal or near-optimal solution. Since then, a variety of scheduling problems are solved using Hopfield type networks (Chen & Dong, 1999; Foo, Takefuji, & Szu, 1995; Liansheng, Gang, & Shuchun, 2000; Lo & Bavarian, 1993; Satake, Morikawa, &

Nakamura, 1994; Vaithyanathan & Ignizio, 1992; Willems & Brandts, 1995; Zhou, Cherkassy, Baldwin, & Olson, 1991).

But a few papers are proposed for the solution of parallel machine scheduling problem using ANNs. Park, Kim, & Lee (2000) presented a backpropagation network for solving identical parallel machine scheduling problems with sequence dependent set up times. They tried to find the sequence of jobs processed on each machine with the objective of minimizing weighted tardiness. Hamad, Sanugi, & Salleh (2003) dealt with the non-identical parallel machines problem with the sum of earliness and tardiness cost minimization and proposed a way of representing the problem to be fed into a backpropagation network. Akyol &Bayhan (2005) proposed a coupled gradient network approach for solving the earliness and tardiness scheduling problem involving sequence dependent setup times.

The objective of this research is to apply ANNs to the identical parallel machine scheduling problem for minimizing the makespan. To the best of our knowledge, this study will be the first attempt to solve the considered problem using neural networks. We employ in this chapter a dynamical gradient network approach to attack the problem. After the appropriate energy function is constructed by using a penalty function approach, the dynamics are defined by steepest gradient descent on the energy function. In order to overcome the tradeoff problem encountered in using the penalty function approach, a time varying penalty coefficient methodology is proposed to be used during simulation experiments. We analyze the impact that the initial conditions of the network have on the performance on 5 different data sets by running each data set 20 times for different sizes of jobs and machines.

A general outline of this chapter is as follows. We give a mixed integer programming (MIP) formulation for the identical parallel machine scheduling problem in Section 2, and describe the proposed network in Section 3. Section 4 provides the computational results, and the conclusions with future research directions are given in Section 5.

5.2 Problem Statement

Consider a set J of n jobs J_i ,$i=1,...,n$ to be processed, each of them on one machine, on a set M of m machines M_j , $j=1,...,m$. All the jobs can be processed on any of the m machines. We consider identical machines models, for which the processing times of each job, p_i, are machine independent. The objective is to find an appropriate allocation of jobs to machines that would optimize a performance criterion. We are interested in the makespan criterion (maximum completion time), *Cmax*.

The following notations are used throughout the rest of this chapter.

J_i : job i, i Є $N=\{1,...,n\}$
M_j : machine j, j Є $M=\{1,...,m\}$
p_i: processing time of J_i
C_i: completion time of J_i
C_{max}: makespan, the maximum completion time of all jobs
$C_{max} = \max\{C_1, C_2, ...,C_n\}$
x_{ij} : 0/1 assignment variable = $\begin{Bmatrix} 1 & \textit{if job i is assigned to machine j} \\ 0 & \textit{otherwise} \end{Bmatrix}$

A MIP formulation of the minimum makespan problem can be defined as follows:
min C_{max}
subject to

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad 1 \le i \le n \qquad (5.1)$$

$$C\,\mathbf{max} - \sum_{i=1}^{n} p_i x_{ij} \ge 0 \qquad 1 \le j \le m \qquad (5.2)$$

The first constraint given in (5.1) ensures that each job is assigned to only one machine. The second constraint given in (5.2) ensures that the makespan is at least the completion time of each machine.

5.3 Design of the Proposed Dynamical Gradient Network

In this section, we describe how dynamical gradient networks can be used to solve the considered problem presented in the previous section. The proposed approach is an extension of the original formulation given in Hopfield (1984, 1985). Firstly the network architecture is explained, and then derivation of the energy function representing the proposed network, and dynamics and proof of the convergence of the proposed network are given. Finally, the proposed penalty parameter determination method is illustrated with an example.

5.3.1 The Network Architecture

The proposed gradient network has two types of neurons: a continuous type neuron to represent real valued variable *Cmax*, and discrete types of neurons to represent binary valued variables $X_{11},\ldots, X_{1m};\ X_{21},\ldots, X_{2m};\ X_{n1},\ldots,X_{nm}$. UX_{ij} symbolizes the input to the neuron for job *i* and resource *j*, and *UCmax* denotes the input to the neuron representing *Cmax*. The dynamics of the gradient net will be defined in terms of these input variables.

VX_{ij} designates the output of the neuron for job *i* and resource *j*. This neuron will be activated if job *i* is allocated to resource *j*. *VCmax* depicts the output of the neuron representing *Cmax*. We use a linear type activation function for neuron Cmax. Neurons with sigmoidal nonlinearity are used to represent discrete variables X_{ij}, so that the activation function for discrete neurons can take the usual sigmoidal form with slopes λ_X. Here, we use a log-sigmoid function to convert discrete neurons to continuous ones. Its functional form is shown in Figure 5.2.

5.3.2 The Energy Function

Instead of using linear programming or the *k-out-of-N* rules to develop the energy function, we directly formulate the cost function according to the constraints term by term. The energy function for this network is constructed using a penalty function approach. That is the energy function E consists of the objective function C_{max} plus a penalty function to enforce the constraints. For the problem considered, the penalty function P(X, Cmax) will include three penalty terms: P1, P2 and P3.

The first term P1 adds a positive penalty if the solution does not satisfy any of the equality constraints given in (5.3). In other words, the first term attempts to ensure that each job is allocated to only one machine.

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad 1 \le i \le n \tag{5.3}$$

In this case, P1 $= \sum_{i=1}^{n} (\sum_{j=1}^{m} X_{ij} - 1)^2$. This term yields zero when these equality constraints are satisfied.

P2 adds a positive penalty if the solution does not satisfy any of the inequality constraints given in (5.4).

$$C\mathbf{max} - \sum_{i=1}^{n} p_i x_{ij} \ge 0 \qquad 1 \le j \le m. \tag{5.4}$$

In accordance with this constraint, P2 will take the following form $\sum_{j=1}^{m} v(\sum_{i=1}^{n} p_i X_{ij} - C_{\max})$ where *v* represents the penalty function.

$v(\varepsilon) = \varepsilon^2$ *for all* $\varepsilon > 0$ *and* $v(\varepsilon) = 0$ *for all* $\varepsilon \le 0$ (Watta & Hassoun, 1996), and the functional form of this function is given in Figure 5.1.

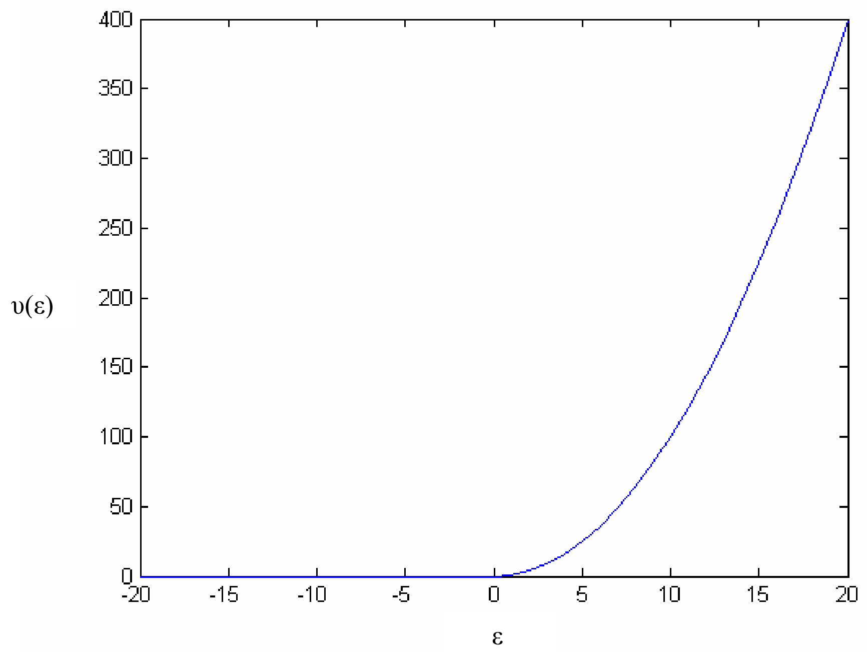


Figure 5.1 Penalty function for enforcing inequality constraints

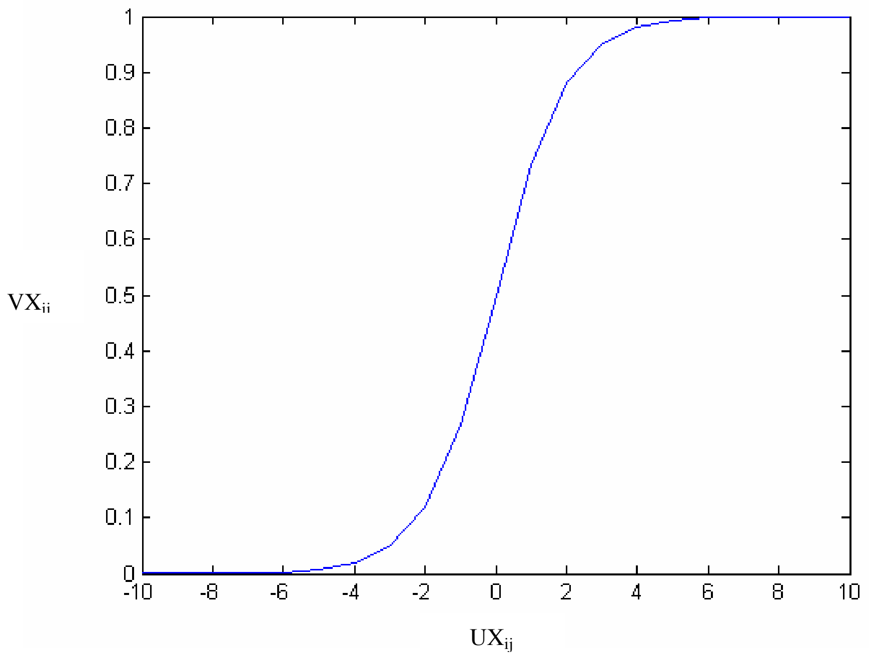


Figure 5.2 Activation function for discrete neurons

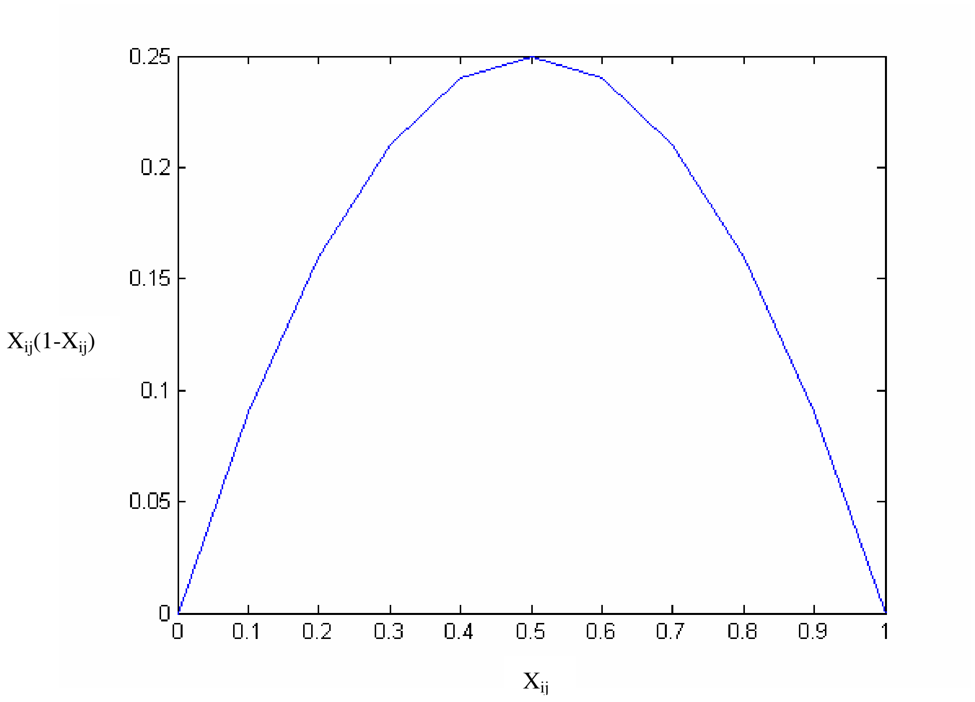


Figure 5.3 Penalty function for enforcing the 0,1 constraints

We require that $X_{ij} \in \{0,1\}$. These constraints will be captured by P3 which adds a positive penalty if the binary constraints $X_{ij} \in \{0,1\}$ are violated. In Figure 5.3, the functional form of this penalty term is shown. It can be seen that the penalty will be zero at either $X_{ij} = 0$ or $X_{ij} = 1$.

P3 $= \sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}(1-X_{ij})$ and correspondingly, the total penalty function P (X, Cmax) with all constraints can be induced as follows.

$$\min\ B\sum_{i=1}^{n}(\sum_{j=1}^{m} X_{ij}-1)^2 + C\sum_{j=1}^{m} v(\sum_{i=1}^{n} p_i X_{ij} - C_{\max}) + D\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}(1-X_{ij})$$

The complete energy function can thus be written as:

$$\min\ AC_{\max} + B\sum_{i=1}^{n}(\sum_{j=1}^{m} X_{ij}-1)^2 + C\sum_{j=1}^{m} v(\sum_{i=1}^{n} p_i X_{ij} - C_{\max}) + D\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}(1-X_{ij})$$

We can write the energy function in terms of output variables as follows:

$$\min\ AVC_{\max} + B\sum_{i=1}^{n}(\sum_{j=1}^{m} VX_{ij} - 1)^2 + C\sum_{j=1}^{m} v(\sum_{i=1}^{n} p_i VX_{ij} - VC_{\max}) + D\sum_{i=1}^{n}\sum_{j=1}^{m} VX_{ij}(1 - VX_{ij})$$

where A, B, C and D are positive penalty coefficients.

5.3.3 The Dynamics

In addition to defining the energy function to be employed, we need to consider the equation of motion of the neuron input. The dynamics for the gradient network are obtained by gradient descent on the energy function. The equations of motion are obtained as follows.

$$\begin{aligned}\frac{dUC_{\max}}{dt} &= -\frac{\partial E}{\partial VC_{\max}} \\ &= -A - (-1)C\sum_{j=1}^{m} \upsilon'[\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max}]\end{aligned} \tag{5.5}$$

$$\begin{aligned}\frac{dUX_{ij}}{dt} &= -\frac{\partial E}{\partial VX_{ij}} \\ &= -2B[\sum_{k=1}^{m} VX_{ik} - 1] - C(Pi)\upsilon'[\sum_{l=1}^{n} p_l VX_{lj} - VC_{\max}] - D(1 - 2VX_{ij})\end{aligned} \tag{5.6}$$

where $\eta_{C\max}$ and η_X are positive coefficients which will be used to scale the dynamics of the network, and ν' is the derivative of the penalty function ν.

$$\upsilon'(\varepsilon) = 2\varepsilon \ \textit{for all}\ \varepsilon \rangle 0 \ \textit{and}\ \upsilon'(\varepsilon) = 0 \ \textit{for all}\ \varepsilon \le 0$$

The computation is performed in all neurons at the same time so that the network operates in a fully parallel mode. The solution of equations of motion may be accomplished via the use of Euler's approximation. The states of the neurons are updated at iteration *k* as follows.

$$UC_{\max}{}^{k} = UC_{\max}{}^{k-1} + \eta_{C\max}\frac{dUC\max}{dt} \tag{5.7}$$

$$UX_{ij}{}^{k} = UX_{ij}{}^{k-1} + \eta_{X}\frac{dUX_{ij}}{dt} \tag{5.8}$$

Neuron outputs are calculated by V=g (U), where g (.) is the activation function, U is the input and V is the output of the neuron.

VCmax=g(UCmax) = UCmax (a linear function)

VX_{ij} = g(UX_{ij}) = logsig ($\lambda_X \times UX_{ij}$) (a log-sigmoid function)

where λ_X is the slope of the activation function and logsig(n) = 1 / (1 + exp(-n)).

5.3.4 Proof of Convergence

In order to use the proposed Hopfield-like dynamical network for the solution of the problem, we have to prove the convergence of the network. To do so, we have to show that energy does not increase along the trajectories, energy is bounded below, the solutions are bounded and time derivative of the energy is equal to zero only at equilibria.

Consider the time derivative of the energy function E given below.

$$\begin{aligned}\frac{dE}{dt} &= \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\partial E}{\partial VX_{ij}}\frac{\partial VX_{ij}}{dt} + \frac{\partial E}{\partial VC\max}\frac{\partial VC\max}{dt}\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} -\frac{dUX_{ij}}{dt}\frac{dVX_{ij}}{dt} + \frac{\partial E}{\partial VC\max}\frac{\partial VC\max}{dt}\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}\left(-\frac{dVX_{ij}}{dt}\right)\frac{dUXij}{dVX_{ij}}\left(\frac{dVX_{ij}}{dt}\right) + \left(\frac{-dUC\max}{dt}\right)\frac{dUC\max}{dt}\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} -\left(\frac{dVX_{ij}}{dt}\right)^{2}\frac{dUX_{ij}}{dVX_{ij}} - \left(\frac{dVC\max}{dt}\right)^{2}\end{aligned} \tag{5.9}$$

Since $\frac{dUX_{ij}}{dVX_{ij}} = [g^{-1}(VX_{ij})]' \geq 0$ (monotone increasing) for log-sigmoid function, the right-hand side of the equation given in (5.9) will be obviously negative. This ensures that the energy does not increase along trajectories, so we can write $\frac{dE}{dt} \leq 0$.

$\frac{dE}{dt} = 0$ implies that $\frac{dVX_{ij}}{dt} = 0$ *for all* i, j and $\frac{dVC\max}{dt} = 0$. In other words, $\frac{dE}{dt} = 0$ at the equilibrium points.

Since X$_{ij}$s are binary variables, they are bounded but we have to check the boundedness of Cmax. If we rewrite the motion equation for Cmax, we obtain the following:

$$\frac{dUC_{\max}}{dt} = -\frac{\partial E}{\partial VC_{\max}}$$
$$= -A - (-1)C\sum_{j=1}^{m} \upsilon'[\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max}]$$

There may be different possible cases

Case 1: Assume that $\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} = 0 \rightarrow \frac{dUC_{\max}}{dt} = -A$

which means that UCmax=VCmax will decrease. This will cause

$\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} > 0$.

Case 2: Assume that $\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} < 0 \rightarrow \frac{dUC_{\max}}{dt} = -A$

which means that UCmax=VCmax will decrease. This will cause

$\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} > 0$.

Therefore we have to consider Case 3 in which we assume $\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} > 0$

If $\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max} > 0$

$$\frac{dUC_{\max}}{dt} = \frac{dVC_{\max}}{dt} - \frac{\partial E}{\partial VC_{\max}}$$

$$= -A - (-1)C\sum_{j=1}^{m} \upsilon'[\sum_{i=1}^{n} P_i VX_{ij} - VC_{\max}]$$

$$= -A - 2CVC_{\max} + 2C\sum_{i=1}^{n} P_i VX_{ij}$$

Let r(z)$= -A + 2C\sum_{i=1}^{n} P_i VX_{ij}$

$$\frac{dVC_{\max}}{dt} = -2CVC_{\max} + r(z)$$

If we multiply both sides by e$^{\text{Ct,}}$

we get $e^{Ct} \cdot \frac{dVC_{\max}}{dt} = -2CVC_{\max} \cdot e^{Ct} + r(z) \cdot e^{Ct}$

$$e^{Ct} \cdot \frac{dVC_{\max}}{dt} + 2CVC_{\max} \cdot e^{Ct} = r(t) \cdot e^{Ct}$$

$$= \frac{d}{dt}[e^{Ct} VC_{\max}] = r(t)e^{Ct}$$

$$\int_{VC\max(0)e^{0.C}}^{VC\max(t)e^{Ct}} d(VC_{\max}(t).e^{Ct}) = \int_0^t e^{Cz} r(z)dz$$

$$= VC_{\max}(t)e^{Ct} - VC_{\max}(0) = \int_0^t e^{Cz} r(z)dz$$

$$= VC_{\max}(t) = e^{-Ct} VC_{\max}(0) + \int_0^t e^{-Ct} e^{Cz} r(z)dz$$

$$= e^{-Ct} VC_{\max}(0) + e^{-Ct} \int_0^t e^{Cz} r(z)dz$$

$$|VC_{\max}(t)| = \left| e^{-Ct} VC_{\max}(0) + e^{-Ct} \int_0^t e^{Cz} r(z)dz \right|$$

$$\leq \left| e^{-Ct} \cdot VC_{\max}(0) \right| + \left| e^{-Ct} \cdot \int_0^t e^{Cz} r(z)dz \right|$$

$$\leq e^{-Ct} \left| VC_{\max}(0) \right| + e^{-Ct} \cdot \left| \int_0^t e^{Cz} r(z)dz \right|$$

We can write

$$|VC_{\max}(t)| \le e^{-Ct}|VC_{\max}(0)| + e^{-Ct} \cdot \left| \int_0^t e^{Cz} r(z) dz \right|$$

$$\le e^{-Ct}|VC_{\max}(0)| + e^{-Ct} \cdot \int_0^t |e^{Cz}||r(z)| dz$$

Assume that $|r(z)| \le M < \infty$

$$|VC_{\max}(t)| \le |e^{-Ct}||VC_{\max}(0)| + e^{-Ct} \cdot M \int_0^t e^{Cz} dz$$

$$\le e^{-Ct}|VC_{\max}(0)| + e^{-Ct} \cdot M \cdot \frac{1}{c}[e^{Ct} - 1]$$

$$\le e^{-Ct}|VC_{\max}(0)| + \frac{M}{c}[1 - e^{-Ct}]$$

Since $|e^{-Ct}| \le 1$ and $e^{-Ct} \to 0$ as $t \to \infty$, then

$|VC_{\max}(t)| \le \infty$

$$r(z) = -A + 2C\sum_{i=1}^{n} P_i VX_{ij}$$

$$|r(z)| = \left| -A + 2C\sum_{i=1}^{n} P_i VX_{ij} \right|$$

$$\le |-A| + \left| 2C\sum_{i=1}^{n} P_i VX_{ij} \right|$$

$$\le A + 2C\sum_{i=1}^{n} P_i VX_{ij}$$

Since A>0 and $2C\sum_{i=1}^{n} P_i VX_{ij} > 0$

$|r(z)| < \infty$ and $\frac{dUC_{\max}}{dt} < \infty$

and we can conclude that the solutions are bounded.

Combining this fact with the fact that the energy E is bounded (since the cost is always greater than zero), we conclude that the network converges to a stable state which is a local minimum of E(X,Cmax). In other words, the time evolution of the network is a motion in space tends to that minimum point as *t* goes to infinity.

5.3.5 Selection of the Parameters

In order to simulate the proposed network for solving the problem described by the dynamics given in Section 5.3.3, some parameters should be determined. These are the penalty parameters A, B, C and D; the activation slopes $\lambda_{X;}$ the step sizes η_{Cmax}, η_X and the initial conditions.

Because there is no theoretically established method for choosing the values of the penalty coefficients for an arbitrary optimization problem, the appropriate values for these coefficients can be determined empirically. That is simulation runs are conducted, and optimality and/or feasibility of the resulting equilibrium points of the system are observed. The network can be initialized to small random values, and then synchronous or asynchronous updating of the network will allow a minimum energy state to be attained. In order to ensure smooth convergence, step size must be selected carefully (Watta & Hassoun, 1996).

The dynamics of the proposed Hopfield-like gradient network will converge to local minima of the energy function E. Since the energy function includes four terms, competing to be minimized, there are many local minima and a tradeoff among the terms. An infeasible solution may be obtained when at least one of the constraint penalty terms is non-zero. In this case, the objective function term will generally be quite small but the solution will not be feasible. Alternatively, a local minimum, which causes a feasible but not a good solution, may be encountered even if all the constraints are satisfied. In order to satisfy the each penalty term, its associated penalty parameter can be increased but this results an increase in other penalty terms and a tradeoff occurs. The penalty parameters that result a feasible and a good solution, which minimizes the objective function, should be found.

Determining the appropriate values of the penalty parameters, network parameters and initial states are so critical issues associated with gradient type networks that by adjusting the parameters, the convergence performance to valid solutions can be improved. It is clear that solving scheduling problems represented by many constraints will cause a tradeoff among the penalty terms to be minimized.

Due to the problems of Hopfield like NNs in solving optimization problems, various modifications are proposed to improve the convergence of the Hopfield network. While several authors modified the energy function of the Hopfield network to improve the convergence to valid solutions (Aiyer, Niranjan, & Fallside, 1990; Brandt, Wang, Laub & Mitra, 1988; Van Den Bout & Miller, 1988), many others studied the same formulation with different penalty parameters (Hedge, Sweet, & Levy, 1988; Kamgar-Parsi & Kamgar-Parsi, 1992; Lai & Coghill, 1992). In recent years, time based penalty parameters are proposed to overcome the tradeoff problems encountered in using penalty function approach. Wang (1991) used monotonically time-varying penalty parameters for solving convex programming problems. Dogan & Guzelis (2006) proposed linearly increasing time-varying penalty parameters for solving clustering problems. Here, we propose to use time varying penalty parameters that take zero values as a starting value and then are increased in a linear fashion in a stepwise manner to reduce the feasible region and also by updating all the neurons synchronously, better simulation results are obtained.

The proposed gradient network algorithm can be summarised by the following pseudo-code.

Step 1. Construct an energy function for the considered problem using a penalty function approach.

Step 2. Initialize all neuron states to random values.

Step 3. Select the slope of the activation function (λ) and step sizes (η).

Step 4. Determine penalty parameters

Step 4.1 Select C (the coefficient of the inequality constraint) and assign zero as initial value to other penalty parameters A, B and D. If the constraint associated with parameter C is satisfied, proceed to Step 4.2 otherwise go back to Step 4.1.

Step 4.2 Select D (a higher value than C to increase the effect of equality constraint), and use the predetermined value of C (without taking into consideration of the effect of parameter A and B) to check whether both of the constraints associated with these terms are satisfied. If yes go to step 4.3, otherwise to step 4.4.

Step 4.3. Select B (a higher value than D), assign 1 to A, and use the predetermined values of C, D together with B to check whether all of the constraints associated with these terms are satisfied. If yes go to step 5, otherwise to step 4.4.

Step 4.4. Increase the value of parameter whose associated constraint is not satisfied.

Step 5. Repeat n times:

Step 5.1. Update U using equations (5.7) and (5.8), and then compute V by V=g (U).

Step 6. If the energy has converged to local minimum proceed to step 7, otherwise go back to step 5.

Step 7. Examine the final solution to determine feasibility and optimality.

Step 8. Adjust parameters A, B, C, D if necessary to obtain a satisfactory solution, reinitialize neuron states and repeat from step 5.

5.3.6 An example

We explain the procedure with a 5-job 3-machine identical parallel machine scheduling problem. After constructing the energy function for this problem, all neuron states are initialized to random values chosen uniformly from the interval [0,1]. In the proposed approach, we firstly suggest to satisfy the inequality constraint by penalizing it. In the first phase of the simulation (for the first 2000 iterations), initial value of the penalty parameter C is chosen as 8. Because other penalty parameters are not taken into consideration, they are equal to zero. Since this inequality constraint is satisfied after 2000 iterations, it is decided to proceed to the next phase. In the second phase (for iterations from 2001 to 4000), one of the equality constraints (binary constraints) is taken into consideration, and its associated parameter D is chosen as 20, a value greater than C. The predetermined value of C, 8, is used to penalize the inequality constraint. Both of the constraints are satisfied. Thus, it is decided to proceed to the next phase (for iterations from 4001 to 5000). In this phase, all of the constraints are tried to be satisfied. Together with the predetermined values of C and D, the penalty parameter B belonging to the assignment constraint is chosen as 100 (a value greater than other parameters). Since A belongs to the original objective function, it is not penalized, and we assign 1 to A. After running simulations with all these 4 penalty terms, the feasibility and optimality of the final solution is checked. It is seen that except the inequality constraint, being violated with a small percentage error, all of the constraints are satisfied. Therefore, it is decided to enhance the weight of this constraint, and then value of its parameter, C, is increased to 600. Optimal solution is found at iteration 5100. All of the constraints were met satisfactorily, and the cost value is 3.1. In Table 5.1, values of penalty parameters used during the solution of the problem considered are displayed. Figure 5.4 illustrates the evolution of the energy of the network during simulation and the penalty parameter values in four phases of simulation. As it is shown in this figure, in each phase of the simulation, the values of the parameters that result a cost value of zero is tried to be found. By increasing the penalty coefficient of each constraint in a stepwise manner, the feasibility region is aimed to be narrowed. Since all the constraints are taken into consideration during iterations

4001 to 5000, at the beginning of the simulation, the cost value increases from zero to higher values. But after an oscillation process the states of the network converge, however, a small cost value of 0.03 comes from the violation of the inequality constraint whose satisfaction ensures that the makespan is at least the completion time of each machine. Therefore, by only penalizing this inequality with a high value of 600, an optimal solution is obtained.

Table 5.1 Penalty parameter values in four phases of simulation

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

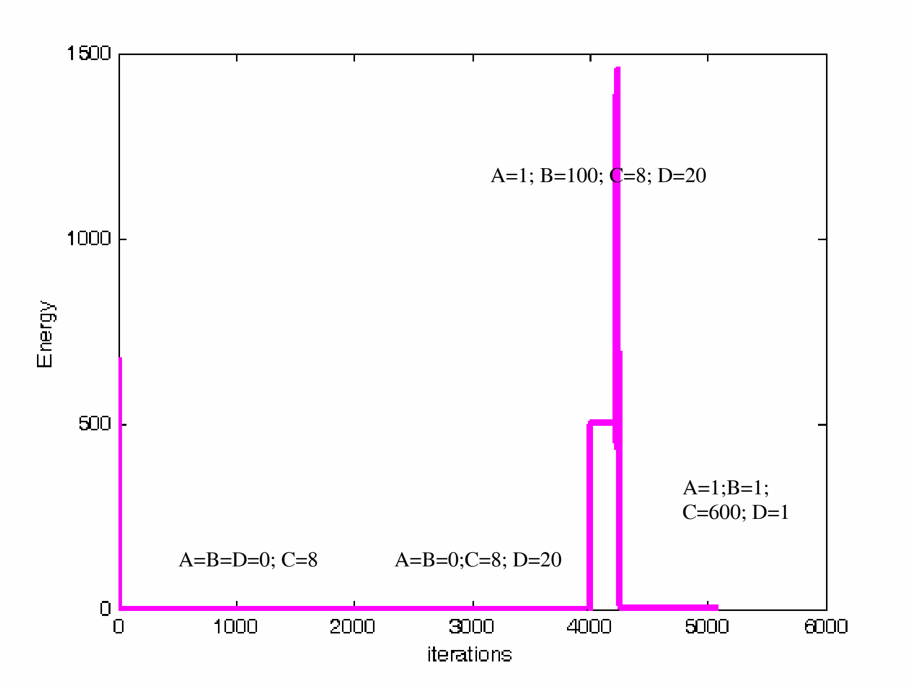


Figure 5.4 Energy evolution during simulation

5.4 Simulation Results

A simulation experiment was conducted to test the effectiveness of the proposed gradient network approach in terms of solution quality. The initial conditions of the network and the processing times of jobs were chosen randomly from uniform distributions on [0,1], and [1,3], respectively. In tables 5.2-5.12, penalty coefficients of the proposed gradient network and in table 5.13, other parameters which were determined empirically by running trial simulations are given, respectively.

For each problem size, the gradient network was run for 20 different initial conditions on 5 different datasets. It is to be noted that the same set of penalty parameters are tried to be found for all the test sets of each problem size during simulations. By tuning the parameters for each dataset, it is possible to improve the performance of the proposed network.

Table 5.2 Penalty coefficients during four phases of simulations for n=5, m=3

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 5.3 Penalty coefficients during four phases of simulations for n=10, m=3

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 5.4 Penalty coefficients during four phases of simulations for n=20, m=3

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 5.5 Penalty coefficients during four phases of simulations for n=50, m=3

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 5.6 Penalty coefficients during four phases of simulations for n=100, m=3

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 5.7 Penalty coefficients during four phases of simulations for n=500, m=3

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	8	0
2001:4000	0	0	8	80
4001:5000	1	450	8	80
5001:5100	1	1	500	1

Table 5.8 Penalty coefficients during four phases of simulations for n=10, m=5

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	8	0
2001:4000	0	0	8	20
4001:5000	1	100	8	20
5001:5100	1	1	600	1

Table 5.9 Penalty coefficients during four phases of simulations for n=20, m=5

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	10	0
2001:4000	0	0	10	30
4001:5000	1	100	10	30
5001:5100	1	1	600	1

Table 5.10 Penalty coefficients during four phases of simulations for n=50, m=5

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	10	0
2001:4000	0	0	10	30
4001:5000	1	100	10	30
5001:5100	1	1	600	1

Table 5.11 Penalty coefficients during four phases of simulations for n=100, m=5

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	10	0
2001:4000	0	0	10	30
4001:5000	1	100	10	30
5001:5100	1	1	600	1

Table 5.12 Penalty coefficients during four phases of simulations for n=500, m=5

Penalty Coef. / *Iterations*	*A*	*B*	*C*	*D*
1:2000	0	0	8	0
2001:4000	0	0	8	70
4001:5000	1	300	8	70
5001:5100	1	1	400	1

Table 5.13 Other Parameters used in the simulation

m	n	η_{Cmax}	η_X	λ_X
3	5	0.001	0.1	1
3	10	0.001	0.1	1
3	20	0.001	0.1	1
3	50	0.001	0.1	1
3	100	0.001	0.1	1
3	500	0.0008	0.1	1
5	10	0.001	0.01	1
5	20	0.0008	0.01	1
5	50	0.0008	0.1	1
5	100	0.0008	0.1	1
5	500	0.0008	0.1	1

The proposed procedure was implemented in Matlab language (Version 6.5) and run on a PC with a Pentium IV, 2.6 GHz processor having a 512 MB of RAM. In tables 5.14-5.24, the solutions obtained by the gradient network using the determined parameters are compared with those of the well known LPT heuristic and with the optimum solutions found by Lingo (version 8.0), a linear programming software package, in terms of Best Cmax (cost of the best solution obtained by the gradient network), Avg. Cmax (cost of the average solution obtained by the gradient network), Worst Cmax (cost of the worst solution obtained by the gradient network), and % deviations. In these tables, columns (6) and (7) represent the % deviations of the proposed gradient network solution from the LPT rule solution and from the optimal solution, respectively. The % deviations are given by

$$\%\ deviation\ from\ LPT = \frac{Avg.\,C\max(Gradient\ network) - C\max(LPT)}{C\max(LPT)} * 100\%$$

$$\%\ deviation\ from\ the\ optimal = \frac{Avg.\,C\max(Gradient\ network) - C\max(optimal)}{C\max(optimal)} * 100\%$$

where Avg. Cmax(Gradient network) is the average gradient network solution of the 20 runs, Cmax(LPT) is the LPT solution, and Cmax(optimal) is the optimal solution obtained by the linear programming solver. The percentage of times, which resulted in a feasible solution by the network, was also displayed in the last columns of these tables. It is obvious that the negative % deviation values from the LPT dispatching rule represent the % improvement realized by the gradient network.

As our primary goal was to compare the proposed network solution with the LPT rule and with the optimal solutions, in terms of solution quality, the CPU times required for solving each data set are not given. But from the simulation experiments, it is seen that when compared with the very long solution times needed to obtain the optimal solutions by the Lingo software, the proposed network could converge to valid solutions in reasonable times in 13.18 seconds for n=3, m=5 and in 406.27 seconds for n=500 m=5. Obviously, by the implementation of parallel neural processing, significant reductions can be obtained in computational time.

Table 5.14 Results for m=3, n=5 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
3.1	3.1	3.1	3.1	3.1	0.00	0.00	100%
4.69	4.69	4.69	4.69	4.69	0.00	0.00	100%
3.55	3.55	3.55	3.55	3.55	0.00	0.00	100%
2.98	2.98	2.98	2.98	2.98	0.00	0.00	100%
3.02	3.02	3.02	3.02	3.02	0.00	0.00	100%

Table 5.15 Results for m=3, n=10 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
7.33	7.54	7.67	7.59	7.21	-0.66	4.57	100 %
6.97	7.21	7.47	7.45	6.92	-3.22	4.19	100 %
7.28	7.56	7.72	7.69	7.2	-1.69	5.00	100 %
6.79	7.11	7.30	7.46	6.72	-4.69	5.80	100 %
6.77	7.01	7.31	7.44	6.72	-5.78	4.31	100 %

Table 5.16 Results for m=3, n=20 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
13.24	13.53	13.85	13.37	13.05	1.19	3.68	100 %
13.84	14.24	14.46	14.01	13.74	1.64	3.64	100 %
13.03	13.42	13.63	13.40	12.92	0.15	3.87	100 %
14.25	14.54	14.76	14.60	14.05	-0.41	3.48	100 %
13.35	13.60	13.82	13.46	13.12	1.04	3.66	100 %

Table 5.17 Results for m=3, n=50 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
33.53	33.84	34.07	33.70	33.34	0.41	1.50	100 %
30.58	30.95	31.14	30.75	30.36	0.65	1.94	100 %
31.47	31.85	32.15	31.65	31.38	0.63	1.49	100 %
34.53	35.41	35.77	35.32	34.92	0.25	1.40	100 %
34.68	35.10	35.30	34.88	34.51	0.63	1.71	100 %

Table 5.18 Results for m=3, n=100 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
70.00	70.28	70.58	70.55	69.91	-0.38	0.53	100 %
66.65	66.94	67.14	67.09	66.45	-0.22	0.73	100 %
68.42	68.85	69.10	69.04	68.39	-0.27	0.67	100 %
66.11	66.73	66.52	66.73	66.09	0.00	0.97	100 %
65.85	66.15	66.33	66.35	65.69	-0.30	0.70	100 %

Table 5.19 Results for m=3, n=500 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
338.22	338.22	338.22	338.51	338.18	-0.0856	0.012	95 %
329.80	330.59	331.28	330.11	329.78	0.145	0.245	100 %
332.91	333.80	334.66	332.91	332.58	0.267	0.367	100 %
330.02	331.06	332.07	330.28	329.95	0.236	0.336	100 %
332.45	333.48	334.43	332.78	332.45	0.21	0.309	100 %

Table 5.20 Results for m=5, n=10 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
3.43	3.53	3.68	3.43	3.43	2.91	2.91	100 %
3.38	3.76	3.97	3.79	3.38	-0.79	11.24	100 %
3.64	3.85	3.97	3.68	3.57	4.35	7.56	100 %
4.03	4.16	4.24	4.03	4.03	3.22	3.22	100 %
3.57	3.67	3.73	3.53	3.53	3.97	3.97	100 %

Table 5.21 Results for m=5, n=20 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
7.43	7.78	7.91	7.37	7.28	5.56	6.87	100 %
7.68	7.95	8.08	7.62	7.49	4.33	6.14	100 %
8.13	8.24	8.37	7.8	7.76	5.64	6.18	100 %
7.79	7.98	8.13	7.69	7.51	3.77	6.26	100 %
8.55	8.77	8.92	8.29	8.18	5.79	7.21	100 %

Table 5.22 Results for m=5, n=50 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
20.49	20.86	21.09	20.28	20.22	2. 86	3.16	100 %
21.70	22.17	22.42	21.55	21.49	2.88	3.16	100 %
18.69	18.94	19.15	18.42	18.40	2.82	2.93	100 %
20.71	21.11	21.33	20.37	20.33	3.63	3.83	100 %
19.79	20.01	20.24	19.43	19.41	2.98	3.09	100 %

Table 5.23 Results for m=5, n=100 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
41.65	41.87	42.06	41.24	41.20	1.53	1.63	100 %
40.16	40.56	40.74	39.78	39.77	1.96	1.99	100 %
41.90	42.12	42.28	41.36	41.34	1.84	1.89	100 %
40.20	40.55	40.69	39.83	39.82	1.80	1.83	100 %
41.54	41.89	42.06	41.19	41.15	1.70	1.8	100 %

Table 5.24 Results for m=5, n=500 over 5 problems

Gradient Network			LPT (4)	Optimum (5)	Percent Deviation from the LPT solution (6)	Percent Deviation from the optimal solution (7)	Percent Feasibility of Computed Solutions (8)
Best Cmax (1)	Avg. Cmax (2)	Worst Cmax (3)					
199.54	200.15	200.61	200.77	198.97	-0.3088	0.59	95 %
199.54	200.02	200.33	200.77	199.04	-0.37	0.49	100 %
206.42	206.69	207.34	206.33	204.58	0.174	1.03	100 %
197.14	198.18	198.8	198.62	196.85	-0.22	0.676	100 %
194.56	195.32	196.38	195.58	193.86	-0.13	0.753	100 %

To interpret the findings in a table, Table 5.14 is considered. For all the 5 data sets, 20 out of the 20 runs of the proposed network resulted in a feasible solution, hence percent feasibility is 100 %. The average, worst and the best cost of the 20 feasible solutions for the first dataset are equal to the value of the global optimal solution, 3.1. Similarly, if we consider Table 5.23, for the first dataset, again, 100 % of the runs conducted by the proposed network resulted in a feasible solution. The average Cmax of the feasible solutions is 41.87. It is 1.53 % more costly than the result of LPT rule, and 1.63 % more costly than the global optimal solution. The best makespan value produced by the gradient network is 41.65, which is 0.99 % ([(41.65-41.24)*100]/41.24) above the LPT result and 1.09 % ([(41.65-41.20)*100]/41.20) above the global optimal solution.

According to these findings, it is clear that the initial conditions of the network appear to have a serious impact on the solution quality. For example in Table 5.20, for n=10 and m=5, although the proposed network results in gaps between 2.91 and 4.35 % from the LPT solution, on average, it outperforms the LPT heuristic for one of the datasets. In the same table, if the results obtained using the first data set are considered, it is seen that although the average makespan from the 20 different initial runs is found as 3.53, the best makespan out of the 20 runs, produced by the proposed network is equal to the optimal solution, 3.43. In addition, although the average Cmax results obtained by the proposed network are above the LPT results for the 4 data sets, the best Cmax results outperform the LPT rule in 4 data sets.

In all the simulations carried out to show the performance of the network, convergence to valid schedules is achieved and better results are obtained for small number of machines and large number of jobs. If all the test cases are considered, the proposed network is, on average, able to produce a solution with a makespan value, which is 1.07 % above the LPT result. By tuning the penalty coefficients for each dataset, it is possible to improve the convergence and the optimality of the solutions. On the other hand, besides its convergence to valid schedules, convergence to good quality solutions of the proposed network points out its general applicability in other scheduling environments.

5.5 Conclusions

This study has presented a dynamical gradient network for solving the identical parallel machine scheduling problem with the makespan criterion which is known to be NP-hard even for the case of two identical parallel machines. Focus of this chapter has been on demonstrating the optimization capabilities of the proposed network by solving a set of randomly generated problems. The proposed Hopfield-like network uses time-varying penalty parameters that start from zero and increase in a stepwise manner during iterations to overcome the tradeoff problem of the penalty function method, one of the important drawbacks of the penalty function approach. To analyze the performance of the network, it is compared with the well-known LPT heuristic commonly used to solve the problem under study, and also with the optimal solutions in terms of the solution quality. The simulation experiments demonstrated that the proposed network generated feasible solutions in all the cases, and in some of the data sets it found smaller makespan compared to LPT. In general, for all the instances, the average deviation percentage of the proposed network is 1.07 % above the LPT heuristic.

By conducting several simulation experiments, the influence of different initializations schemes was investigated on the solutions of the problem considered. The analysis results showed that the percent error of the network is very sensitive to the selection of the starting points and the choice of the parameters used in simulation.

The contribution of this study is two fold. We propose to use a novel time varying penalty method that guarantees feasible and near optimal solutions for solving the identical parallel machine scheduling problem with the makespan criterion. Although a large body of literature exists for solving identical parallel machine scheduling problem with the makespan minimization criterion, to the best of our knowledge, there is no previously published article that tried to solve this NP-hard problem using neural networks. Therefore, this study will also make a contribution to the scheduling literature.

Several issues are worthy of future investigations. First, further studies will be focused on selecting the parameters of the network automatically rather than choosing by trial and error, which is one of the drawbacks of neural networks. Second, extension of the results to large size problems will be worthwhile. Finally, extension of the results to different manufacturing scheduling environments is important for industrial applications, and implementation of the network in hardware can make progress in computational efficiency.

CHAPTER SIX
SOLUTION of MULTI-MACHINE EARLINESS AND TARDINESS SCHEDULING PROBLEM USING AN INTERCONNECTED NEURAL NETWORK APPROACH

6.1 Introduction

With the successful implementation of the Just-in-Time concept in inventory/production management in today's manufacturing environments, it is needed to complete the jobs as close as possible to their due dates. In other words, finishing jobs earlier than the due dates is considered as undesirable as finishing jobs late. Therefore, both early and tardy completion of a job with respect to its due date is penalized. If the jobs are completed earlier than their due dates, an earliness penalty will be incurred which can be regarded as a holding cost for finished goods, deterioration of perishable goods and opportunity costs. If the jobs are completed later than their due dates, a tardiness penalty which can be regarded as the backlogging cost including the cost of customer compensation for missing the due date or the loss of goodwill will be incurred. These costs motivated researchers to consider both earliness and tardiness (E/T) as penalties in the objective function of a schedule. In recent years, majority of scheduling studies on E/T have dealt with single machine scheduling problems. While Kanet (1981), Sundararaghavan & Ahmed (1984), Hall (1986), Bagchi, Sullivan, & Chang (1986) studied single machine models with common due dates for all jobs, Abdul-Razaq & Potts (1988), Ow & Morton (1988), Ow & Morton (1989) considered distinct due dates for each job.

The single machine E/T problem introduced by Kanet (1981) and Sidney (1977), in its simplest form, is to schedule N jobs on a single machine to minimize the weighted differences between job completion times and due dates. Since then, many researchers worked on various extensions of the problem. For a detailed survey of the earlier applications, refer to Rachavachari (1988) and Baker & Scudder (1990).

However, the E/T literature becomes scarce when we consider the problem of scheduling jobs on multiple machines. Sundararaghavan & Ahmed (1984), Arkin & Roundy (1991), De, Ghosh, & Wells (1994) are known to be the first researchers dealing with the problem of scheduling *N* jobs on *M* identical parallel machines with the objective of minimizing the total penalty costs for E/T.

Heady & Zhu (1998) studied the problem of scheduling *N* jobs, with sequence dependent setup times, on identical parallel machines to minimize the sum of weighted E/T, and proposed a heuristic algorithm to solve it. For the same objective, Sivrikaya-Serifoglu & Ulusoy (1999) presented two different genetic algorithm approaches to solve the problem of scheduling jobs with sequence dependent setup times on two types of parallel machines; uniform and identical. Balakrishan, Kanet, & Sridharan (1999) proposed a compact mixed integer formulation for scheduling jobs with sequence dependent setup times on uniform parallel machines, and evaluated its performance on small sized problems. In another study, Radhakrishnan & Ventura (2000) employed simulated annealing to schedule jobs with sequence dependent setup times on identical parallel machines to obtain near optimal solutions. Following this work, Sun & Wang (2003) proposed a dynamic programming algorithm and two heuristics to minimize the total weighted earliness and tardiness for identical parallel machine scheduling problem.

In this chapter, we study the problem of scheduling *N* jobs with non-common due dates and sequence-dependent set-up times on *M non-identical* machines. This is an NP-hard problem since the special case with a single machine is even known to be NP-hard (Garey, Tarjan, & Wilfong, 1988). In most of the machine scheduling researches sequence-dependent set-up times were not considered, and the set-up times were assumed to be sequence independent and to be a part of the job processing times. Scheduling problems with E/T measures of performance in the presence of sequence-dependent set-up times are mathematically complex to solve, and optimal solutions cannot be obtained even for problems of reasonable size, therefore heuristics have to be utilized to provide good near optimal solutions. Despite a large number of approaches existing in the literature, to find an efficient method to obtain optimal solutions in polynomial time motivated the researchers to apply neural networks as a promising approach to scheduling problems and over the last decades, there has been an explosion of interest in using Artificial Neural Networks (ANNs) for the solution of various scheduling problems. Most of the scheduling problems are solved using Hopfield-like networks. However, by performing gradient descent on the energy function, the Hopfield model gets easily trapped in local minimum states, and this causes decreasing efficiency especially in large sized problems. Additionally, determining the appropriate values of the penalty parameters, network parameters and initial states are other critical issues associated with this model.

To lessen the burden of computations for determining the proper values for the penalty parameters and to obtain feasible results, Takefuji, Lee, & Aiso (1992) and Lee, Funabiki, & Takefuji (1992) introduced a neural network called the maximum neural network. This network is based on a competitive Hopfield-type network. The significant advantage of using the maximum neural network is that it does not demand fine-tuning of parameters, as most Hopfield networks do. In this network, which is composed of groups of neurons, a competitive winner-take-all rule is imposed for updating the neurons. Thus, the neuron with the maximum input per group is the only one that has nonzero output. This model has shown to provide powerful approaches for combinatorial optimization problems (Lee, Funabiki, & Takefuji, 1992; Lee & Takefuji, 1992; Takefuji, 1992; Funabiki, Takenaka, & Nishikawa, 1997) and for polygonal approximation (Chung, Tsai, Chen, & Sun, 1994).

Recently, new maximum neural models are proposed by Galan-Marin & Munoz Perez (2001) for the *n*-queens and the bipartite subgraph problems, and by Galan-Marin, Merida-Casermeiro, & Munoz-Perez (2003) for solving the maximum clique problem. From the literature reviewed, it is seen that neural networks seldom include competitive architecture into the network for solving the scheduling problems.

The objective of this research is to utilize ANNs to tackle the problem of scheduling a set of independent jobs including sequence dependent setup times, on non-identical multiple machines to minimize the total weighted earliness and tardiness. Here, by non-identical machines, we mean unrelated machines, where each job on each machine has a different processing time and complicates the problem. In addition, the due dates of the jobs are distinct which complicates the problem further. To the best of our knowledge, there are no published articles in the literature that tried to solve this problem except the study of Zhu & Heady (2000) where they provided a mixed integer programming formulation to represent the problem. Thus, this study will be the first attempt to solve the problem considered using neural networks. We employ in this chapter a dynamical gradient network approach to attack the problem. The proposed Hopfield-like system is composed of two maximum neural networks, three piecewise linear and one log-sigmoid network, all of which are connected to each other. The aim of using maximum networks is to reduce the network complexity and to obtain a simplified energy function. After the appropriate energy function was constructed by using a penalty function approach, the dynamics are defined by steepest gradient descent on the energy function and the proposed approach is illustrated through the case of an 8 jobs to be processed on 3 machines in a JIT manufacturing environment. The rest of the chapter is organized as follows. In Section 2, a mixed integer programming formulation is presented for the problem addressed in this chapter. Section 3 describes the proposed coupled network. Section 4 discusses the computational experience and the proposed penalty determination process. Finally, Section 5 concludes the chapter with directions for future work

6.2 Problem Statement and Formulation

In this section, we deal with the problem of job scheduling on non-identical multiple machines to minimize the total penalty costs for earliness and tardiness with sequence dependent setup times. The problem includes non-common due dates and non-uniform cost penalties. We are given N independent jobs $J = \{J_1,\dots,J_N\}$ to be scheduled on M non-identical machines where $N \geq M$.

We use the notation of *N/M/ET* with sequence-dependent setups to designate this problem.

The notation for the problem:

β: a large number
d_i: due date for job *i*
e_i: earliness cost for job *i*
E_i: Earliness of job *i*
p_{im}: processing time for job i using machine *m*
s_{ji}: setup time for job *i* when it immediately follows job *j*
s_{0i}: setup time for job *i* when it is the first in queue
t_i: tardiness cost for job *i*
T_i: Tardiness of job *i*
X_i: completion time of job *i*
Y_{ijm}: 1 if job *i* precedes job *j* on machine m, 0 otherwise
Z_{im}: 1 if job *i* is processed on machine *m,* 0 otherwise

The multi-machine earliness-tardiness problem can be formulated by using the following mixed integer programming (MIP) model (Zhu & Heady, 2000).

Objective function:

$$\min \quad \sum_{i=1}^{N} (e_i E_i + t_i T_i) \tag{6.1}$$

Subject to:

$$X_i - T_i + E_i = d_i \qquad i = 1,...,N \tag{6.2}$$

$$\sum_{m=1}^{M} Z_{im} = 1 \qquad i = 1,...,N \tag{6.3}$$

$$\sum_{j=1}^{N} Y_{ijm} \le Z_{im} \qquad j \ne i, i = 1,...,N, \quad m = 1,...M \tag{6.4}$$

$$\sum_{i=0}^{N} Y_{ijm} = Z_{jm} \qquad j \neq i, j = 1,...,N, \quad m = 1,...M \tag{6.5}$$

$$X_i - X_j - \beta Y_{jim} \geq p_{im} + s_{ji} - \beta \quad j \neq i, i = 1,...,N, j = 0,1,...,N \quad m = 1,...,M \tag{6.6}$$

$$\sum_{j=1}^{N} Y_{ojm} = 1 \qquad m = 1,...,M \tag{6.7}$$

$$Z_{im} \in \{0,1\} \qquad i = 1,...,N; m = 1,...,M \tag{6.8}$$

$$Y_{ijm} \in \{0,1\} \qquad i = 0,...,N; j = 1,...,N; m = 1,...,M \tag{6.9}$$

$$X_i, T_i, E_i \geq 0 \qquad i = 1,...,N \tag{6.10}$$

where the processing times on each job p_{im} are deterministic and known in advance, preemption of jobs is not allowed. Let X_i be the completion time and d_i the due date of job *i*. Job *i* is early if $X_i<d_i$; it is tardy if $X_i>d_i$; and it is on time if $X_i=d_i$. The earliness and tardiness of job *i* are defined by $E_i= \max(0; d_i-X_i)$ and $T_i= \max(0; X_i-d_i)$, respectively.

All decision variables are non-negative and Y_{ijm} and Z_{im} are binary valued variables. It is assumed that a dummy *job 0* which is always at the first position on each machine is present. Obviously, we can write $Z_{0m}=1$ and $X_0 = 0$. The objective function to be minimized given in (6.1) was built by the sum of cost-weighted deviations in job completion times from the job due dates. The first constraint defines the relationship between the completion time, the due date, and earliness and tardiness of each job. The second constraint states that each job is processed on one and only one machine. The third and fourth constraints ensure that each job (but not the last scheduled job) must come immediately before, and each job (but not the first scheduled job) must come immediately after, only one other job. The fifth constraint guarantees that the completion time of job *i* is far enough after that of job *j* to include the processing time and setup time for job *i*. By the inclusion of constraint given in (6.7), the set-up time for the real job assigned to the first position in the sequence on each machine will be taken into consideration. While the constraints given in (6.8) and (6.9) correspond to the integrality constraints, the last constraint given in (6.10) imposes the variables to be positive.

6.3 Design of the Proposed Interconnected Neural Network

In this section, we describe how the proposed dynamical network can be used to solve the considered problem presented in the previous section. The proposed approach is an extension of the original formulation given in Hopfield & Tank (1985) and Hopfield (1984). Firstly, the network architecture is explained, and then it is followed by the derivation of the energy function representing the proposed network. Then, the dynamics are obtained and the convergence of the proposed network is discussed. Finally, the proposed network is explained with an example.

6.3.1 The Network Architecture

The proposed coupled gradient network consists of six interconnected sub-networks: two maximum neural networks; an $N\times M$, Z and a $1\times N\times M$ YO network, three $N\times 1$ piecewise linear; namely E, T and X networks and one $N\times N\times M$ log-sigmoid Y network where N and M are the number of jobs and the number of machines, respectively. One of the maximum neural networks (Z network) is used to assign each job on only one machine and the other one (YO network) to assign a dummy *job 0* at the beginning of the sequence before all the real jobs on each machine. Three piecewise linear networks called the E, T and X networks are used to represent continuous variables E_i, T_i and X_i for $i=1,\dots,N$. The log-sigmoid network Y is used to represent binary valued variables, Y_{ijm} for $i=1,\dots,N; j=1,\dots,N; m=1,\dots,M$.

The input-output scheme for each of the neural sub-networks is shown in Figure 6.1. The input to the *i*th node will be denoted by UE_i in the E network, by UT_i in the T network, by UX_i in the X network, by UZ_{im} for the (i,m)th neuron in the Z network, UY_{0jm} for the (j,m)th neuron in the YO network, and by UY_{ijm} for the (i,j,m)th neuron in the Y network. The dynamics of the coupled net will be defined in terms of these input variables.

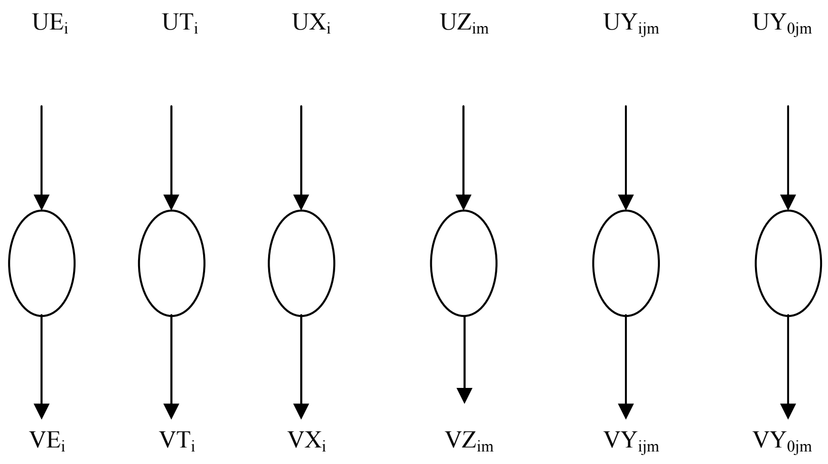
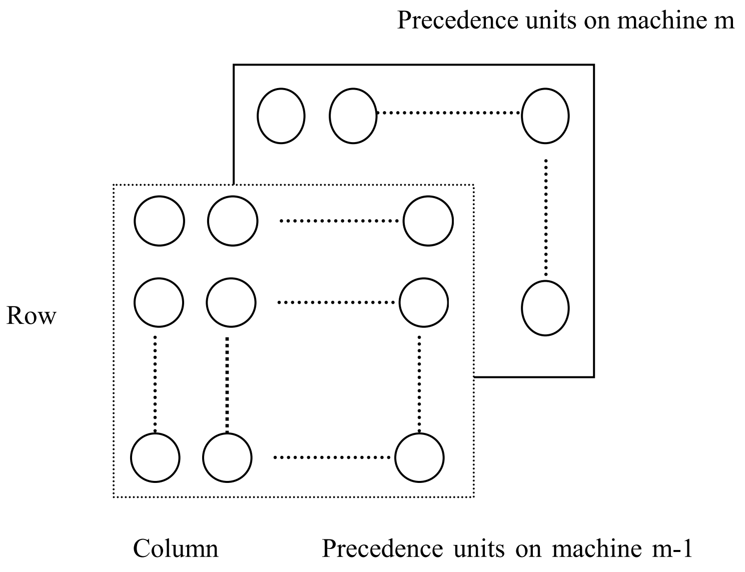


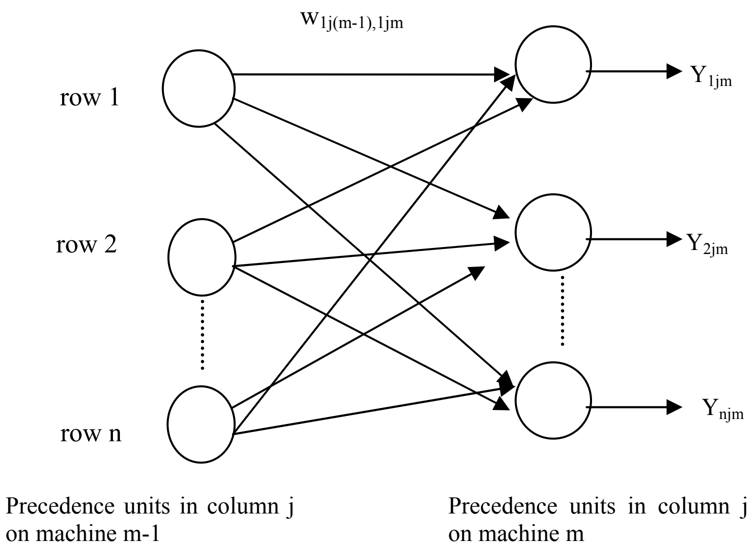
Figure 6.1 The input-output scheme for the neurons representing each unit

Similarly, node outputs (states) of the E, T, X, and Z networks will be the variables $VE_1, VE_2, \ldots, VE_N$; $VT_1, VT_2, \ldots, VT_N$; $VX_1, VX_2, \ldots, VX_N$; and $VZ_{11}, VZ_{12}, \ldots, VZ_{NM}$, respectively. It is to be noted that rather than using network Y to represent variables VY_{0jm}, here, we use network YO to represent the dummy jobs at the beginning of the sequence before job *j* on machine *m.* Therefore, the variables VY_{0jm} for $j=1,2,..,N$; $m=1,2,...,M$, and VY_{ijm} for $i=1,2,...,N$; $j=1,2,...,N$; and $m=1,2,...,M$ will demonstrate node outputs of the YO network and Y network, respectively. Figure 6.2 given below shows the arrangement of the precedence units and Figure 6.3 depicts the connections of the precedence units, Y_{ijm} on different machines. Other types of units are arranged in similar ways, but are not represented as multidimensional arrays.



Precedence units on machine m
Row
Column
Precedence units on machine m-1

Figure 6.2 Arrangement of precedence units



$w_{1j(m-1),1jm}$
row 1
row 2
row n
Y_{1jm}
Y_{2jm}
Y_{njm}
Precedence units in column j on machine m-1
Precedence units in column j on machine m

Figure 6.3 Connections of precedence units Y_{ijm} on different machines

6.3.2 The Energy function

The energy function for this network is constructed using a penalty function approach. That is, the energy function *E* consists of the objective function

$\sum_{i=1}^{N}(e_i E_i + t_i T_i)$ plus a penalty function P(E, T, X, Z,Y, YO) to enforce the constraints. The penalty function involves the sum of the penalty terms each of which corresponds to each constraint of the problem.

The first penalty term, P1=$\sum_{i=1}^{N}(X_i - T_i + E_i - d_i)^2$, will add a positive penalty if the solution does not satisfy any of the equality constraints given in (6.2), and will yield zero when these equality constraints are satisfied.

To prevent the assignment of each job on more than one machine, the second penalty term, P2=$\sum_{i=1}^{N}(\sum_{m=1}^{M} Z_{im} - 1)^2$, which will add a positive penalty if the solution does not satisfy any of the equality constraints given in (6.3), is included in the energy function.

The third penalty term, P3, will add a positive penalty if the solution does not satisfy any of the inequality constraints given in (6.4). In accordance with this constraint, P3 will take the following form, P3=$\sum_{i=1}^{N}\sum_{m=1}^{M} v(\sum_{j=1, i\neq j}^{N} Y_{ijm} - Z_{im})$, where v represents the penalty function. $v(\varepsilon) = \varepsilon^2$ *for all* $\varepsilon > 0$ *and* $v(\varepsilon) = 0$ *for all* $\varepsilon \le 0$ and the functional form of this function is shown in Figure 6.4.

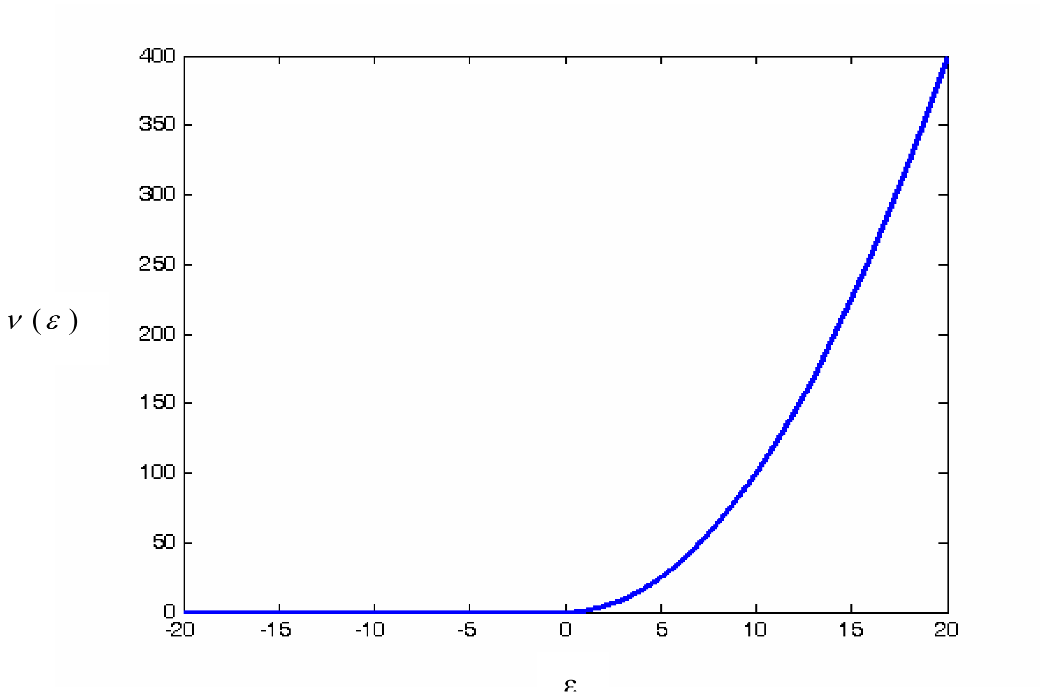


Figure 6.4 Penalty function for enforcing inequality constraints

The fourth term P4 will add a positive penalty if any of the equality constraints given in (6.5) is violated. Therefore, P4 should be defined by

$$\text{P4} = \sum_{j=1}^{N}\sum_{m=1}^{M}\Big(\sum_{i=0,i\neq j}^{N} Y_{ijm} - Z_{jm}\Big)^2 .$$

The fifth penalty term, P5, is responsible for satisfying the inequality constraints given in (6.6). Therefore it will yield zero when all these inequalities are satisfied. P5 may be written as

$$\text{P5}= \sum_{i=1,i\neq j}^{N}\sum_{j=0}^{N}\sum_{m=1}^{M} \nu(X_j - X_i + \beta(Y_{jim} - 1) + p_{im} + s_{ji}) .$$

To deal with the introduction of a dummy *job 0* at the beginning of the sequence before all the real jobs on each machine, the sixth penalty term *P6* is to be defined as

$$P6 = \sum_{m=1}^{M}\Big(\sum_{j=1}^{N} Y_{0jm} - 1\Big)^2 .$$

This term will add a positive penalty if any of the equality constraints given in (6.7) is violated.

We require that Y_{ijm} and Z_{im} $\epsilon\{0,1\}$. These constraints will be captured by the seventh and the eighth terms, *P7* and *P8*, which will add a positive penalty if the binary constraints given in (6.8) and (6.9) are violated. Hence,

$$P7 = \sum_{i=1}^{N}\sum_{m=1}^{M} Z_{im}(Z_{im}-1) \text{ and}$$

$$P8 = \sum_{i=0}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M} Y_{ijm}(1-Y_{ijm}) = \sum_{j=1}^{N}\sum_{m=1}^{M} Y_{0jm}(1-Y_{0jm}) + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M} Y_{ijm}(1-Y_{ijm})$$

The non-negativity constraints given in (6.10) are not added to the energy function as penalty terms since these constraints will be captured by using an input-output function, *g*, where $g(\varepsilon) = \varepsilon \; for\, all \; \varepsilon \geq 0 \, and \;\; g(\varepsilon) = 0 \; for \; all \; \varepsilon < 0$. Its functional form is given in Figure 6.5. In other words, for zero and positive input values, the activation function will be linear, and so the outputs will be equal to the inputs of the neurons, and for the negative values the output values will be zero.

Therefore, the penalty function for the coupled gradient network can be written as follows:

$$\begin{aligned}
P = {} & B\sum_{i=1}^{N}(X_i - T_i + E_i - d_i)^2 + C\sum_{i=1}^{N}(\sum_{m=1}^{M} Z_{im} - 1)^2 \\
& + D\sum_{i=1}^{N}\sum_{m=1}^{M} v(\sum_{j=1, i\neq j}^{N} Y_{ijm} - Z_{im}) + E\sum_{j=1}^{N}\sum_{m=1}^{M}(\sum_{i=0, i\neq j}^{N} Y_{ijm} - Z_{jm})^2 \\
& + F\sum_{i=1}^{N}\sum_{j=0}^{N}\sum_{m=1}^{M} v(X_j - X_i + \beta(Y_{jim} - 1) + p_{im} + s_{ji}) + G\sum_{m=1}^{M}(\sum_{i=1}^{N} Y_{0im} - 1)^2 \\
& + H\sum_{i=1}^{N}\sum_{m=1}^{M} Z_{im}(1 - Z_{im}) + I\sum_{j=1}^{N}\sum_{m=1}^{M} Y_{0jm}(1 - Y_{0jm}) + J\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M} Y_{ijm}(1 - Y_{ijm}) \qquad (6.11)
\end{aligned}$$

If we sum the objective function given in (6.1) and the penalty function, we will have the following energy function to be minimized:

$$\begin{aligned}
&A\sum_{i=1}^{N}(e_iE_i + t_iT_i) + B\sum_{i=1}^{N}(X_i - T_i + E_i - d_i)^2 + C\sum_{i=1}^{N}(\sum_{m=1}^{M}Z_{im} - 1)^2 \\
&+ D\sum_{i=1}^{N}\sum_{m=1}^{M}\nu(\sum_{j=1,i\neq j}^{N}Y_{ijm} - Z_{im}) + E\sum_{j=1}^{N}\sum_{m=1}^{M}(\sum_{i=0,i\neq j}^{N}Y_{ijm} - Z_{jm})^2 \\
&+ F\sum_{i=1}^{N}\sum_{j=0}^{N}\sum_{m=1}^{M}\nu(X_j - X_i + \beta(Y_{jim} - 1) + p_{im} + s_{ji}) + G\sum_{m=1}^{M}(\sum_{i=1}^{N}Y_{0im} - 1)^2 \\
&+ H\sum_{i=1}^{N}\sum_{m=1}^{M}Z_{im}(1 - Z_{im}) + I\sum_{j=1}^{N}\sum_{m=1}^{M}Y_{0jm}(1 - Y_{0jm}) + J\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}Y_{ijm}(1 - Y_{ijm}) \qquad (6.12)
\end{aligned}$$

where *A,B,C,D,E,F,G,H,I* and *J* are positive penalty coefficients.

If we rewrite the energy function in terms of the output variables, we may obtain

$$\begin{aligned}
&E(VE,VT,VX,VZ,VY,VYO) = A\sum_{i=1}^{N}(e_iVE_i + t_iVT_i) + B\sum_{i=1}^{N}(VX_i - VT_i + VE_i - d_i)^2 \\
&+ C\sum_{i=1}^{N}(\sum_{m=1}^{M}VZ_{im} - 1)^2 + D\sum_{i=1}^{N}\sum_{m=1}^{M}\nu(\sum_{j=1,i\neq j}^{N}VY_{ijm} - VZ_{im}) + E\sum_{j=1}^{N}\sum_{m=1}^{M}(\sum_{i=0,i\neq j}^{N}VY_{ijm} - VZ_{jm})^2 \\
&+ F\sum_{i=1}^{N}\sum_{j=0}^{N}\sum_{m=1}^{M}\nu(VX_j - VX_i + \beta(VY_{jim} - 1) + p_{im} + s_{ji}) + G\sum_{m=1}^{M}(\sum_{j=1}^{N}VY_{0jm} - 1)^2 \\
&+ H\sum_{i=1}^{N}\sum_{m=1}^{M}VZ_{im}(1 - VZ_{im}) + I\sum_{j=1}^{N}\sum_{m=1}^{M}VY_{0jm}(1 - VY_{0jm}) + J\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}VY_{ijm}(1 - VY_{ijm}) \qquad (6.13)
\end{aligned}$$

Applying a *winner take all* (WTA) mechanism to the network, the energy terms with weighting factors *C*, *G*, *H* and *I* can be omitted from the energy function. The *WTA* learning rule guarantees the satisfaction of Eq. (6.3), that is, assignment of each job to only one machine. In addition, it ensures the binary constraint $Z_{im} \in \{0,1\}$.

Similarly, by imposing WTA rule, the constraint $\sum_{j=1}^{N}Y_{ojm} = 1 \quad \forall m = 1,...,m$ and the binary constraint $Y_{0jm} \in \{0,1\}$ will also be satisfied. The energy terms for these constraints are also dropped from the energy function. By this way, these energy terms will be handled explicitly. Therefore, the energy function takes the following form:

$$E(VE,VT,VX,VZ,VY,VYO) = A\sum_{i=1}^{N}(e_i VE_i + t_i VT_i) + B\sum_{i=1}^{N}(VX_i - VT_i + VE_i - d_i)^2$$
$$+ D\sum_{i=1}^{N}\sum_{m=1}^{M}\nu(\sum_{j=1,i\neq j}^{N} VY_{ijm} - VZ_{im}) + E\sum_{j=1}^{N}\sum_{m=1}^{M}(\sum_{i=0,i\neq j}^{N} VY_{ijm} - VZ_{jm})^2$$
$$+ F\sum_{i=1}^{N}\sum_{j=0}^{N}\sum_{m=1}^{M}\nu(VX_j - VX_i + \beta(VY_{jim} - 1) + p_{im} + s_{ji}) + J\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M} VY_{ijm}(1 - VY_{ijm})$$

The penalty term $J\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M} VY_{ijm}(1 - VY_{ijm})$ can also be eliminated from the energy function because these integrality constraints may be satisfied by using a sigmoidal type activation function for variables Y_{ijm} in obtaining the output values. Final form of the energy function can be written as follows.

$$E(VE,VT,VX,VZ,VY,VYO) = A\sum_{i=1}^{N}(e_i VE_i + t_i VT_i) + B\sum_{i=1}^{N}(VX_i - VT_i + VE_i - d_i)^2$$
$$+ D\sum_{i=1}^{N}\sum_{m=1}^{M}\nu(\sum_{j=1,i\neq j}^{N} VY_{ijm} - VZ_{im}) + E\sum_{j=1}^{N}\sum_{m=1}^{M}(\sum_{i=0,i\neq j}^{N} VY_{ijm} - VZ_{jm})^2$$
$$+ F\sum_{i=1}^{N}\sum_{j=0}^{N}\sum_{m=1}^{M}\nu(VX_j - VX_i + \beta(VY_{jim} - 1) + p_{im} + s_{ji}) \tag{6.14}$$

Although the original energy function given by (6.13) includes many penalty terms to be minimized using a difficult trial-and error procedure, by imposing a competitive WTA rule for the updating of the neurons, we get rid of the trouble of determining the proper values for some of the weighting factors. We can see from the above equation that except the weighting factor of the original objective function, the resulting energy function includes only 4 penalty parameters to be determined.

6.3.3 The Dynamics

The dynamics for the coupled gradient network are obtained by gradient descent on the energy function. The motion equations for the neurons are obtained as follows:

For the E network

$$\frac{dUE_i}{dt} = -\frac{\partial E}{\partial VE_i} = -[Ae_i + 2B(VE_i + VX_i - VT_i - d_i) \tag{6.15}$$

For the T network

$$\frac{dUT_i}{dt} = -\frac{\partial E}{\partial VT_i} = -\left[At_i + 2B(VT_i - VX_i - VE_i + d_i)\right] \tag{6.16}$$

For the X network

$$\frac{dUX_i}{dt} = -\frac{\partial E}{\partial VX_i} = -\left[\begin{array}{l} 2B(VX_i + VE_i - d_i - VT_i) \\ + F\sum_{j=0}^{N}\sum_{m=1}^{M} - v'(VX_j - VX_i + \beta(VY_{jim} - 1) + s_{ji} + p_{im}) \\ + F\sum_{l=1,l\neq i}^{N}\sum_{m=1}^{M} v'(VX_i - VX_l + \beta(VY_{ilm} - 1) + s_{il} + p_{lm}) \end{array}\right] \tag{6.17}$$

For the Z network

$$\frac{dUZ_{im}}{dt} = -\frac{\partial E}{\partial VZ_{im}} = -\left[2C(\sum_{m=1}^{M} VZ_{im} - 1) - Dv'(\sum_{j=1,i\neq j}^{N} VY_{ijm} - VZ_{im}) + 2E(VZ_{im} - \sum_{j=0,i\neq j}^{N} VY_{jim})\right] \tag{6.18}$$

For the Y network

$$\frac{dUY_{ijm}}{dt} = -\frac{\partial E}{\partial VY_{ijm}} = -\left[\begin{array}{l} Dv'(\sum_{l=1,i\neq l}^{N} VY_{ilm} - VZ_{im}) + 2E(\sum_{k=0,k\neq j}^{N} VY_{kjm} - VZ_{jm}) \\ + F\beta v'((\beta(VY_{ijm} - 1) + VX_i - VX_j + s_{ij} + p_{jm})) \end{array}\right] \tag{6.19}$$

For the YO network

$$\frac{dUY_{0jm}}{dt} = -\frac{\partial E}{\partial VY_{0jm}} = -\begin{bmatrix} 2E\left[(VY_{0jm}) + \sum_{i=1,i\neq j}^{N} VY_{ijm} - VZ_{jm}\right] \\ + F\beta v'((\beta(VY_{ijm}-1) + VX_i - VX_j + s_{ij} + p_{jm})) \end{bmatrix} \tag{6.20}$$

where ν' is the derivative of the penalty function ν, and $\upsilon'(\varepsilon) = 2\varepsilon \textit{ for all } \varepsilon \rangle 0 \textit{ and } \upsilon'(\varepsilon) = 0 \textit{ for all } \varepsilon \le 0$.

The states of the neurons are updated at iteration *k* by using the first-order Euler method as follows:

$$UE_i^{\,k} = UE_i^{\,k-1} + \eta_E \frac{dUE_i}{dt} \tag{6.21}$$

$$UT_i^{\,k} = UT_i^{\,\mathrm{k-1}} + \eta_T \frac{dUT_i}{dt} \tag{6.22}$$

$$UX_i^{\,k} = UX_i^{\,\mathrm{k-1}} + \eta_X \frac{dUX_i}{dt} \tag{6.23}$$

$$UY_{ijm}^{\,k} = UY_{ijm}^{\,k-1} + \eta_Y \frac{dUY_{ijm}}{dt} \tag{6.24}$$

$$UY_{0jm}^{\,k} = UY_{0jm}^{\,k-1} + \eta_{Y0} \frac{dUY_{0jm}}{dt} \tag{6.25}$$

$$UZ_{im}^{\,k} = UZ_{im}^{\,k-1} + \eta_Z \frac{dUZ_{im}}{dt} \tag{6.26}$$

where η_E, η_T, η_X, η_Z, η_Y and η_{Y0} are positive coefficients which will be used to scale the dynamics of the six networks.

Since the computation is performed in all neurons at the same time, the network operates in a fully parallel mode. Neuron outputs are calculated by V=g (U), where *g* (.) is the activation function, *U* is the input and *V* is the output of a neuron. For the nodes in the E, T and X network, the activation function, *g*, will be given by a piecewise linear function displayed in Figure 6.5, where $g(\varepsilon) = \varepsilon \textit{ for all } \varepsilon \ge 0 \textit{ and } g(\varepsilon) = 0 \textit{ for all } \varepsilon < 0$. In other words,

$VE_i = g(UE_i) = UE_i$ for $UE_i \geq 0$; otherwise $VE_i = 0$ (6.27)

$VT_i = g(UT_i) = UT_i$ for $UT_i \geq 0$; otherwise $VT_i = 0$ (6.28)

$VX_i = g(UX_i) = UX_i$ for $UX_i \geq 0$; otherwise $VX_i = 0$ (6.29)

The activation function for the nodes of the Y network will take the usual sigmoidal form. In other words, continuous relaxation is used for handling discrete variables in the Y network as follows:

$VY_{ijm} = g(UY_{ijm}) = \text{logsig}(\lambda_Y \times UY_{ijm})$ (a log-sigmoid function) (6.30)

where λ_Y is the slope of the activation function and logsig(n) = 1 / (1 + exp(-n)). The functional form of this activation function is given in Figure 6.6.

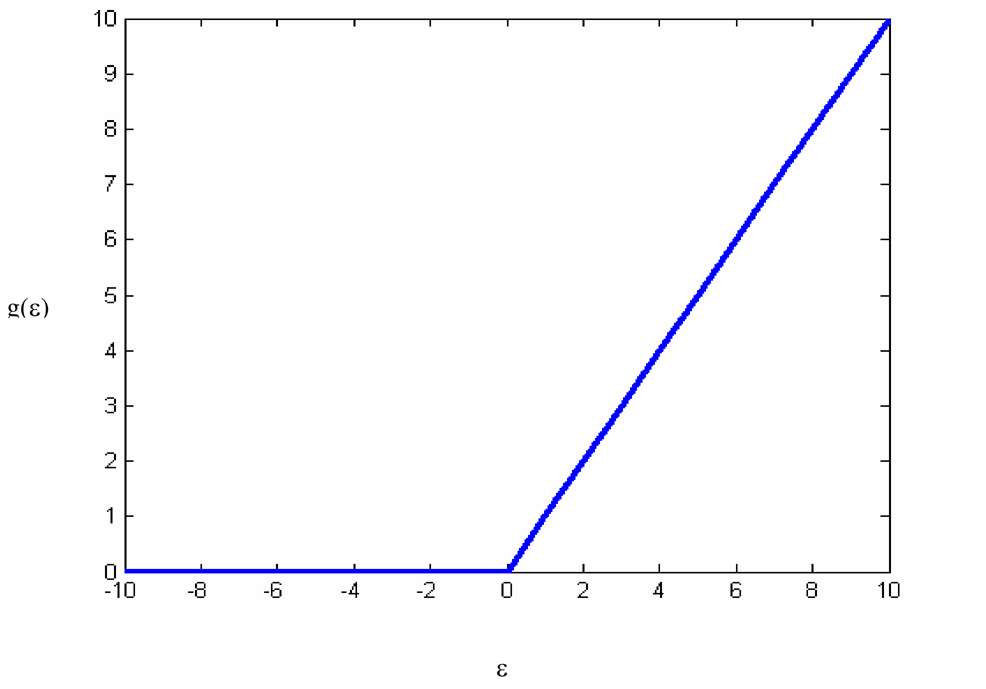


Figure 6.5 Activation function for continuous neurons of E, T and X networks

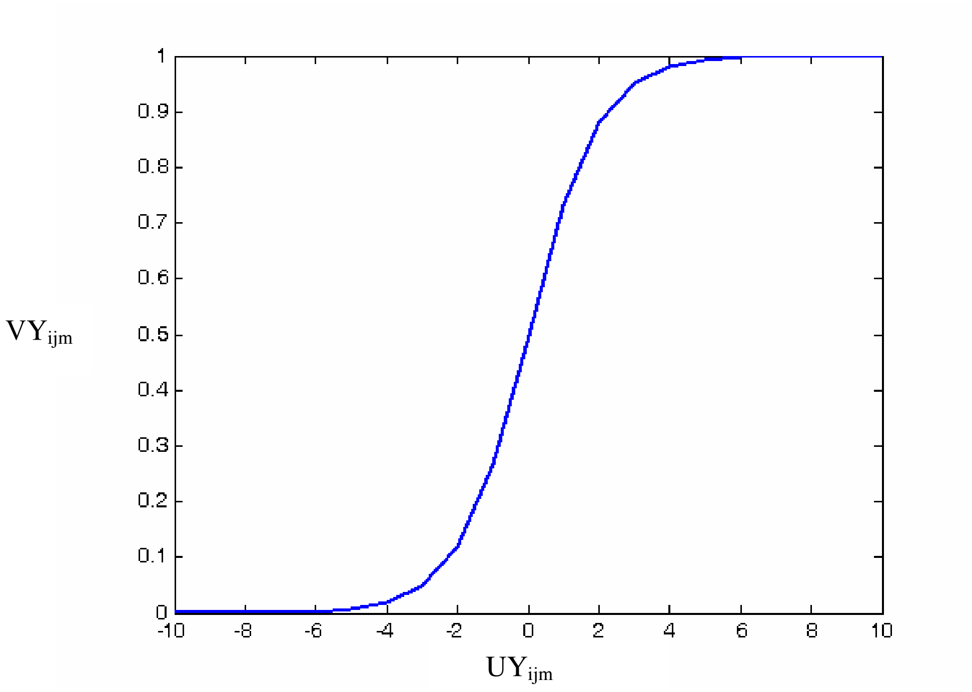


Figure 6.6 Activation function for neurons of the Y network

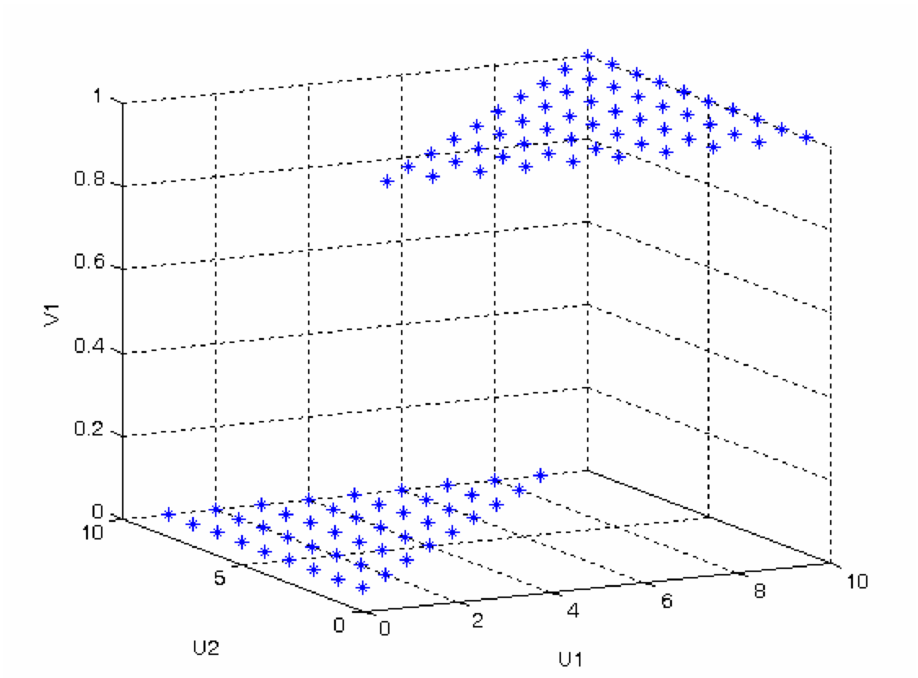


Figure 6.7 Activation function for neurons of the Z network when there are two inputs

The neuron outputs of the Z and YO networks are updated by the maximum neuron model of Takefuji, Lee, & Aiso (1992) as below and its functional form is given in Figure 6.7.

$$VZ_{im} = \begin{cases} 1 & if \quad UZ_{im} = \max(UZ_{i1}, UZ_{i2}, ..., UZ_{im}) \\ 0 & otherwise \end{cases} \tag{6.31}$$

$$VY_{0jm} = \begin{cases} 1 & if \quad UY_{0jm} = \max(UY_{0j1}, UY_{0j2}, ..., UY_{0jm}) \\ 0 & otherwise \end{cases} \tag{6.32}$$

6.3.4 Convergence

In order to use the proposed Hopfield-like network for the solution of the problem, we have to prove the convergence of the network. To do so, we have to show that energy does not increase along the trajectories, energy is bounded below, solutions are bounded and time derivative of the energy is equal to zero only at equilibria. In the proposed network, although VE_is, VT_is, VX_is, VZ_{im}s and VY_{0jm}s are not differentiable functions of time *t*, they have right-hand derivatives.

To prove the convergence of the proposed network, an extension of the La Salle's invariance principle can be used (Sengor, Cakir, Guzelis, Pekergin, & Morgul, 1999). The Lemma below, which is needed for taking the time derivative of the energy, states that the chain rule is valid also for the right derivative.

Definition: The right derivative of a function x(.) : $R \to R^n$ is defined as $\frac{dx(t)}{dt+} := \lim_{\Delta \to 0^+} \frac{x(t+\Delta) - x(t)}{\Delta}$ where $\Delta \to 0^+$ means that Δ approaches zero throughout positive values only (Sengor, Cakir, Guzelis, Pekergin, & Morgul, 1999).

Lemma: Consider the functions $\psi(.) : D_\psi \subset [0, \infty) \to D_g \subset R^n$ *and* $g(.) : D_g \to R$. Let $t \in Int(D_\psi)$ with *Int* stands for the set of interior points. Assume that g(.) is continuously differentiable at *ψ(t)*, and *ψ*(.) is right differentiable at *t*. Then, g o ψ is right differentiable at *t* and $\dfrac{d(g\circ\psi)(t)}{dt^{+}}=[\nabla_{\psi}g(\psi)]\dfrac{d\psi(t)}{dt^{+}}$ (Sengor *et al.*,1999).

Using the Lemma given above, the time derivative of the energy function E can be found as follows:

$$
\begin{aligned}
\frac{dE}{dt^{+}} &= \sum_{i=1}^{N}\frac{\partial E}{\partial VE_i}\frac{dVE_i}{dt^{+}}+\sum_{i=1}^{N}\frac{\partial E}{\partial VT_i}\frac{dVT_i}{dt^{+}}+\sum_{i=1}^{N}\frac{\partial E}{\partial VX_i}\frac{dVX_i}{dt^{+}} \\
&+\sum_{i=1}^{N}\sum_{m=1}^{M}\frac{\partial E}{\partial VZ_{im}}\frac{dVZ_{im}}{dt^{+}}+\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}\frac{\partial E}{\partial VY_{ijm}}\frac{dVY_{ijm}}{dt} \\
&+\sum_{j=1}^{N}\sum_{m=1}^{M}\frac{\partial E}{\partial VY_{0jm}}\frac{dVY_{0jm}}{dt^{+}} \\
&=\sum_{i=1}^{N}\frac{-dUE_i}{dt}\frac{dVEi}{dt^{+}}+\sum_{i=1}^{N}\frac{-dUT_i}{dt}\frac{dVT_i}{dt^{+}}+\sum_{i=1}^{N}\frac{-dUX_i}{dt}\frac{dVX_i}{dt^{+}} \\
&+\sum_{i=1}^{N}\sum_{m=1}^{M}\frac{-dUZ_{im}}{dt}\frac{dVZ_{im}}{dt^{+}}+\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}\frac{-dUY_{ijm}}{dt}\frac{dVY_{ijm}}{dt} \\
&+\sum_{j=1}^{N}\sum_{m=1}^{M}\frac{-dUY_{0jm}}{dt}\frac{dVY_{0jm}}{dt^{+}} \\
&=\sum_{i=1}^{N}\frac{-dUE_i}{dt}\frac{\partial VE_i}{\partial UE_i^{+}}\frac{dUE_i}{dt}+\sum_{i=1}^{N}\frac{-dUT_i}{dt}\frac{\partial VT_i}{\partial UT_i^{+}}\frac{dUT_i}{dt}+\sum_{i=1}^{N}\frac{-dUX_i}{dt}\frac{\partial VX_i}{\partial UX_i^{+}}\frac{dUX_i}{dt} \\
&+\sum_{i=1}^{N}\sum_{m=1}^{M}\frac{-dUZ_{im}}{dt}\frac{\partial VZ_{im}}{\partial UZ_{im}^{+}}\frac{dUZ_{im}}{dt}+\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}\frac{-\partial UY_{ijm}}{\partial VY_{ijm}}\frac{dVY_{ijm}}{dt}\frac{dVY_{ijm}}{dt} \\
&+\sum_{i=1}^{N}\sum_{m=1}^{M}\frac{-dUY_{0jm}}{dt}\frac{\partial VY_{0jm}}{\partial UY_{0jm}^{+}}\frac{dUY_{0jm}}{dt} \\
&= -\sum_{i=1}^{N}\left(\frac{dUE_i}{dt}\right)^2\frac{\partial VE_i}{\partial UE_i^{+}}-\sum_{i=1}^{N}\left(\frac{dUT_i}{dt}\right)^2\frac{\partial VT_i}{\partial UT_i^{+}}-\sum_{i=1}^{N}\left(\frac{dUX_i}{dt}\right)^2\frac{\partial VX_i}{\partial UX_i^{+}} \\
&-\sum_{i=1}^{N}\sum_{m=1}^{M}\left(\frac{dUZ_{im}}{dt}\right)^2\frac{\partial VZ_{im}}{\partial UZ_{im}^{+}}+\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}\frac{-\partial UY_{ijm}}{\partial VY_{ijm}}\left(\frac{dVY_{ijm}}{dt}\right)^2 \\
&-\sum_{i=1}^{N}\sum_{m=1}^{M}\left(\frac{dUY_{0jm}}{dt}\right)^2\frac{\partial VY_{0jm}}{\partial UY_{0jm}^{+}} \qquad (6.33)
\end{aligned}
$$

where $\frac{\partial E}{\partial V}$ is replaced by $\frac{-dU}{dt}$. For the neurons using a piecewise linear function, although the outputs of the neurons are not differentiable functions of time, the right derivative of the outputs with respect to the input variables exists. For example for the neurons of the E network, the time derivative of the neuron output *i*, $\frac{dVE_i}{dt}$ can be written as $\frac{dVE_i}{dt} = \frac{\partial VE_i}{\partial UE_i^+}\frac{dUE_i}{dt}$. Here $\frac{\partial VE_i}{\partial UE_i^+}$ will be equal to one for positive or zero inputs, and will be equal to zero for negative inputs. Therefore, the right derivative of the neuron output represented by VE_i with respect to neuron input UE_i can be written as below.

$$\frac{\partial VE_i}{\partial UE_i^+} = \begin{cases} 1 & if \quad UE_i \geq 0 \\ 0 & otherwise \end{cases}$$

It is equal to heaviside or the unit step function of UE_i represented by U(UEi), and is shown in Figure 6.8.

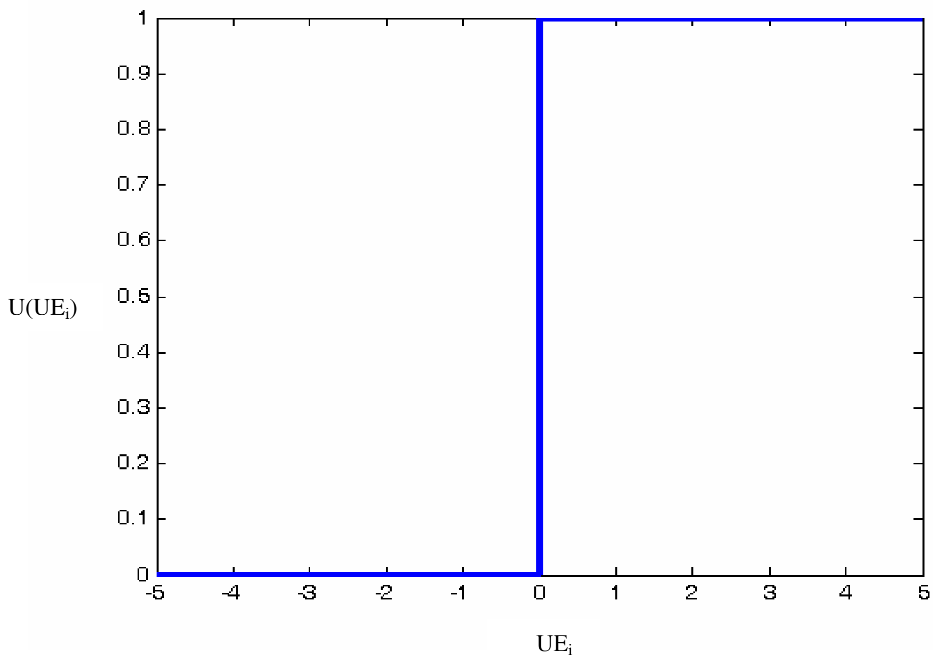


Figure 6.8 Unit step function of UE_i

As the same relation holds for the neurons of T and X networks which are also using a piecewise linear function, we can write

$$\frac{\partial VT_i}{\partial UT_i^+} = U(UT_i) = \begin{cases} 1 & if \quad UT_i \geq 0 \\ 0 & otherwise \end{cases}$$

and

$$\frac{\partial VX_i}{\partial UX_i^+} = U(UX_i) = \begin{cases} 1 & if \quad UX_i \geq 0 \\ 0 & otherwise \end{cases}$$

Therefore, it is obvious that the first three terms in equation (6.33) will be less than or equal to zero.

Similarly, although the outputs of the Z and YO network are not differentiable functions of time, they have right-derivatives, and, for the neurons of the maximum neural network Z, we can write the right-derivative of the energy function with respect to time *t* as follows:

$$\begin{aligned} \frac{dE}{dt^+} &= \sum_{i=1}^{N}\sum_{m=1}^{M} \frac{-dUZ_{im}}{dt}\frac{dVZ_{im}}{dt} \\ &= \sum_{i=1}^{N}\sum_{m=1}^{M} \frac{-dUZ_{im}}{dt}\frac{\partial VZ_{im}}{\partial UZ_{im}^+}\frac{dUZ_{im}}{dt} \end{aligned}$$

Since VZ$_{\text{im}}$s are piecewise constant functions of UZ$_{\text{im}}$s, $\frac{\partial VZ_{im}}{\partial UZ_{im}^+} = 0$. Therefore, the fourth term in equation (6.33) will be zero. For the YO network, we can write

$$\begin{aligned} \frac{dE}{dt^+} &= \sum_{j=1}^{N}\sum_{m=1}^{M} \frac{-dUY_{0jm}}{dt}\frac{dVY_{0jm}}{dt} \\ &= \sum_{j=1}^{N}\sum_{m=1}^{M} \frac{-dUY_{0jm}}{dt}\frac{\partial VY_{0jm}}{\partial UY_{0jm}^+}\frac{dUY_{0jm}}{dt} \end{aligned}$$

In the same fashion, $\dfrac{\partial VY_{0jm}}{\partial UY_{0jm}{}^{+}} = 0$ and the sixth term in (6.33) will be zero.

The fifth term $\displaystyle\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}\frac{-dVY_{ijm}}{dt}\frac{\partial UY_{ijm}}{\partial VY_{ijm}}\frac{dVY_{ijm}}{dt}$ can be written as $\displaystyle\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}-\left(\frac{dVY_{ijm}}{dt}\right)^{2}\frac{dUY_{ijm}}{dVY_{ijm}}$ and since $\dfrac{dUY_{ijm}}{dVY_{ijm}} = [g^{-1}(VY_{ijm})]' \geq 0$ for log-sigmoid function, $\displaystyle\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{M}-\left(\frac{dVY_{ijm}}{dt}\right)^{2}\frac{dUY_{ijm}}{dVY_{ijm}} \leq 0$ so that the right-hand side of the equation given in (6.33) will be obviously negative. Combining this fact with the fact that the energy *E* is bounded below (since the cost is always greater than or equal to zero) we can conclude that the energy does not increase along trajectories, so we can write $\dfrac{dE}{dt^{+}} \leq 0$. All trajectories go to the points where $\dfrac{dE}{dt^{+}} = 0$. Here, it should be noted that if the points are equilibrium points then it can be seen that $\dfrac{dE}{dt^{+}} = 0$.

But for some points, even if $\dfrac{dE}{dt^{+}} = 0$ there is a possibility that $\dfrac{dUE_i}{dt} \neq 0$; $\dfrac{dUT_i}{dt} \neq 0$; $\dfrac{dUX_i}{dt} \neq 0$; $\dfrac{dUZ_{im}}{dt} \neq 0$; $\dfrac{dUY_{0jm}}{dt} \neq 0$ so the trajectories may not reach equilibrium points.

Because VZ_{im}s, VY_{ijm}s and VY_{0jm}s are binary, they will be bounded. But there is no need to check whether VE_is, VT_is and VX_is are bounded or not, since the trajectories may not reach equilibrium points. But this case is never observed during simulation.

6.3.5 Selection of the Parameters

In order to simulate the proposed network for solving the E/T problem described by the dynamics given in Section 6.3.3, some parameters should be determined by trial and error. These are the penalty parameters A, B, D, E and F; the activation slope $\lambda_{Y;}$ the step sizes η_E, η_T, η_X, η_Z, η_Y, $\eta_{YO,}$ and the initial conditions.

Because there is no theoretically established method for choosing the values of the penalty coefficients for an arbitrary optimization problem, the appropriate values for these coefficients can be determined empirically. That is simulation runs are conducted, and optimality and/or feasibility of the resulting equilibrium points of the system are observed. The network can be initialized to small random values, and then synchronous or asynchronous updating of the network may allow a minimum energy state to be attained. In order to ensure smooth convergence, step size must be selected carefully (Watta, 1996).

The dynamics of the proposed Hopfield-like gradient network will converge to local minima of the energy function E. Since the energy function includes five terms, competing to be minimized, there are many local minima and a tradeoff exists among the terms. An infeasible solution may be obtained when at least one of the constraint penalty terms is non-zero. In this case, the objective function term will generally be quite small but the solution will not be feasible. Alternatively, a local minimum, which causes a feasible but not a good solution, may be encountered even if all the constraints are satisfied. In order to satisfy the each penalty term, its associated penalty parameter can be increased. But this causes an increase in other penalty terms and a tradeoff occurs. The penalty parameters that result a feasible and a good solution, which minimizes the objective function, should be found (Smith, 1999).

Due to the problems of Hopfield like NNs in solving optimization problems, various modifications are proposed to improve the convergence of the Hopfield network. While several authors modified the energy function of the Hopfield network to improve the convergence to valid solutions (Aiyer, Niranjan, & Fallside, 1990; Brandt, Wang, Laub, & Mitra, 1988; Van Den Bout & Miller, 1988), many others studied the same formulation with different penalty parameters (Hedge, Sweet, & Levy, 1988; Kamgar-Parsi & Kamgar-Parsi, 1992; Lai & Coghill, 1992). In a different study, Wang (1991) used monotonically time-varying penalty parameters for solving convex programming problems. Here, we propose to use time varying penalty parameters that take zero values as a starting value and then to increase them in a linear fashion step by step to reduce the feasible region, and also to update all the neurons synchronously for obtaining better simulation results.

The proposed gradient network algorithm can be summarized by the following pseudo-code.

Step 1. Construct an energy function for the considered problem using a penalty function approach.

Step 2. Initialize all neuron states to random values.

Step 3. Select the slope of the activation function (λ) and step sizes (η) and determine the penalty parameters evolving with time.

Step 4. Compute the motion equations by (6.15)-(6.20). Update neurons inputs U by the first-order Euler method which is explained through (6.21)-(6.26), and then update the neuron output of variables V using equations (6.27)-(6.32).

Step 5. Repeat the iterations n times and check the cost terms of the energy function penalized. If the required criterion is met, go back to Step 3 to pass to other phase of the simulation. If the work is in the part of the simulation where all the constraints are taken into consideration, check whether the energy has converged to a local minimum. If yes, proceed to step 6 otherwise go back to Step 5.

Step 6. If the energy has converged to local minimum, examine the final solution to determine feasibility and optimality.

Step 7. Adjust parameters A, B, D, E, F if necessary to obtain a satisfactory solution, reinitialize neuron states and repeat from step 3.

6.4 Simulation results

In this section, a simulation experiment is conducted to test the effectiveness of the proposed gradient network approach on an example problem in terms of solution quality. Assume that we are given 8 jobs to be processed on 3 machines in a JIT manufacturing environment. Table 6.1 shows the processing times of each job on each machine. The setup time matrix $\boldsymbol{S}$ represents the setup time incurred between the two jobs. For example S_{12}=1.1 shows the setup time for *job 2* when it immediately follows *job 1*. The setup time vector ***SO*** includes the setup times for each job when they are in the first position. While the due dates are denoted by the elements of vector ***d***, vectors ***e*** and ***t*** depict early and tardy costs for each job.

Table 6.1 Processing times on each machine

Job\Machine	M1	M2	M3
JI	0.53	1.15	0.80
J2	1.19	1.06	0.91
J3	1.12	1.21	0.82
J4	0.98	1.07	0.74
J5	0.63	1.13	1.2
J6	1.14	0.9	0.92
J7	1.26	0.84	0.45
J8	1.38	0.88	0.79

$$\boldsymbol{S} = \begin{bmatrix} 0 & 1.11 & 1.29 & 1.21 & 1.11 & 1.02 & 0.91 & 0.77 \\ 1.23 & 0 & 1.16 & 1.07 & 1.16 & 1.22 & 0.89 & 1.13 \\ 0.91 & 1.05 & 0 & 1.10 & 0.98 & 0.74 & 1.13 & 1.08 \\ 1.45 & 0.88 & 1.05 & 0 & 1.13 & 1.16 & 0.88 & 1.12 \\ 0.98 & 0.92 & 0.85 & 0.91 & 0 & 0.85 & 1.13 & 1.21 \\ 0.56 & 1.10 & 1.09 & 1.30 & 0.69 & 0 & 1.09 & 1.24 \\ 0.82 & 1.17 & 0.59 & 0.89 & 1.13 & 1.14 & 0 & 0.65 \\ 1.22 & 0.95 & 1.14 & 1.08 & 1.16 & 0.73 & 1.27 & 0 \end{bmatrix}$$

SO=[0.44 0.88 0.78 1.09 1.34 0.69 0.9 0.9]

e=[3.31 1.26 0.76 3.74 3.79 4.79 1.12 4.0]

t=[3.31 1.26 0.76 3.74 3.79 4.79 1.12 4.0]

d=[12.95 9.02 13.83 15.03 15.03 11.49 11.44 16.45]

The proposed procedure was implemented in Matlab language (Version 6.5) and the initial conditions of the network were chosen randomly from uniform distribution on the interval [0,1]. A time-varying penalty parameter method is proposed to be used during simulation experiments.

In the following paragraph, we solve the problem applying the steps of the proposed approach given in the previous section.

Step 1. For the problem considered we have the following energy function obtained in (6.14) using a penalty function approach.

$$
\begin{aligned}
E(VE,VT,VX,VZ,VY,VYO) &= A\sum_{i=1}^{N}(e_i VE_i + t_i VT_i) + B\sum_{i=1}^{N}(VX_i - VT_i + VE_i - d_i)^2 \\
&+ D\sum_{i=1}^{N}\sum_{m=1}^{M}\nu\Big(\sum_{j=1,i\neq j}^{N} VY_{ijm} - VZ_{im}\Big) + E\sum_{j=1}^{N}\sum_{m=1}^{M}\Big(\sum_{i=0,i\neq j}^{N} VY_{ijm} - VZ_{jm}\Big)^2 \\
&+ F\sum_{i=1}^{N}\sum_{j=0}^{N}\sum_{m=1}^{M}\nu(VX_j - VX_i + \beta(VY_{jim} - 1) + p_{im} + s_{ji})
\end{aligned}
$$

Step 2. All the neuron inputs are randomly chosen from uniform distribution on the interval [0,1], and the initial values of the neuron outputs are fixed using equations (6.27)-(6.32).

Step 3. For the first phase of the simulation, the activation slope and step sizes are chosen as $\lambda_Y=300$, $\eta_E = 0.001$, $\eta_T = 0.005$, $\eta_X = 0.001$, $\eta_Z = 1.0$, $\eta_Y = 0.0005$, and $\eta_{Y0} = 1$.

We have 5 penalty parameters in the energy function given in (6.14). But, since the penalty parameter *A* belongs to the original objective function, here, we will only deal with the satisfaction of the constraints and try to determine the values of the penalty parameters enforcing the constraints which will guarantee a feasible solution. In the first phase of the simulation, it is decided to penalize both the inequality constraints and the equality constraints by using their associated parameters *D* and *E*. At this first stage, value of zero is assigned to the other parameters *A*, *B* and *F*. In this stage, our aim is to find the values of the penalty parameters *D* and *E* such that the jobs assigned to the first position on each machine represented by the elements of the ***YO*** matrix with values 1, are also assigned to the same position on the job assignment matrix ***Z*****.** We will stop running the iterations when this criterion is met.

Step 4. The motion equations are computed by (6.15)-(6.20) and the neuron inputs U are updated by the first-order Euler method which is explained by the equations through (6.21)-(6.26), and then the neuron output of variables V are updated using equations (6.27)-(6.32).

Step 5. After performing 1000 iterations, it is seen that this criterion is met and the best values of D and E are found as 3 and 0.1, respectively. Since we did not take into consideration all the constraints during this part of the simulation experiment, we continue the procedure starting from Step 3.

In the second stage of the simulation experiment, it is decided to see the impact of the predetermined values of D and E on the results when other constraints and the original objective function are taken into consideration by weighting them with a value of 1. Smaller step sizes are used for updating the neurons of the Z and YO network and a larger step size value is used for updating the neurons of the Y network, which are determined empirically, not to cause a move up to a different region that violates the assignment of the jobs on the first position on each machine to the same machines on the ***Z*** matrix. In this part of the simulation experiment, the activation slope and the step sizes are chosen as $\lambda_Y = 300$, $\eta_E = 0.001$, $\eta_T = 0.005$, $\eta_X = 0.001$, $\eta_Z = 0.001$, $\eta_Y = 0.05$, and $\eta_{Y0} = 1\times10^{-6}$.

Then we proceed to Step 4. The motion equations are computed, neuron inputs U are updated by the first-order Euler method, and the neuron outputs V are updated using equations (6.27)-(6.32)

Proceeding to Step 5, we repeat the iterations n times using the determined parameters. In addition to the 1000 iterations performed in the previous stage, 1000 more iterations are made in this stage. It is seen that although the criterion satisfied in the first stage is not violated, a cost of 21.65 is obtained. This cost implies that the terms, weighted with *A* and *B* in the energy function, are not satisfied.

Since the necessary criterion of the cost function is still met in this stage and no local minimum is obtained, we again go back to step 3.

At the beginning of this step, one important thing to be paid attention is assuring the satisfaction of the binary constraint $Y_{ijm} \in \{0,1\}$, which is included in the energy function. Therefore, the satisfaction of the binary constraints belonging to Y_{ijm}s is checked, and it is seen that some of the Y_{ijm}s are not binary. Since Y_{ijm}s determine the precedence relationships among jobs, the next step should be to penalize the term associated with *F* to satisfy these constraints. Also, to ensure that Y_{ijm}s are binary, a kind of mean field annealing is incorporated into the coupled network by slowly increasing the slope of the activation function during the running of the network. By this way, the slope of the activation function becomes time-varying and can be expressed as $\lambda=\lambda(t)=\lambda_0+\lambda_1 t$ where *t* is the computational time, λ_0 and λ_1 are constants which are determined empirically. Here, in our experiments, we use $\lambda_0=300$, $\lambda_1=0.5$ and *t* represents the iteration number divided by 1500. Therefore, the time varying slope becomes $\lambda(t)= 300+0.5\times$(iteration number/1500). By running trial simulation experiments using penalty terms of 1 for other terms, the best value for *F* that satisfies its related constraints is found as 5. Then we proceed to Step 4 and again compute the motion equations. We update neurons inputs *U* by the first-order Euler method, and then update the neuron output of variables *V*.

In Step 5, 3000 more iterations were needed until all the cost terms are satisfied. Then, the energy function is checked, and since a cost of zero is obtained, work continues from step 6.

In Step 6, we examine the final solution to determine feasibility and optimality. This is the optimum solution, since all the constraints are satisfied and a cost value of zero is obtained.

An optimal set of penalty parameters found for the proposed network are given in Table 6.2. The best values of the step sizes, which were determined empirically and used in all stages of the simulations and the slope values of the logsigmoid activation function, are given in Tables 6.3 and 6.4, respectively.

Table 6.2 Penalty parameter values in three phases of simulation

Iterations\ Penalty Coef.	*A*	*B*	*D*	*E*	*F*
1:1000	0	0	3	0.1	0
1001:2000	1	1	3	0.1	1
2001:5000	1	1	1	1	5

The evolution of the energy during the simulation of the network is given in Figure 6.9. As it is seen from this figure, after an oscillation process, the network converges to an optimal solution where the cost value is zero.

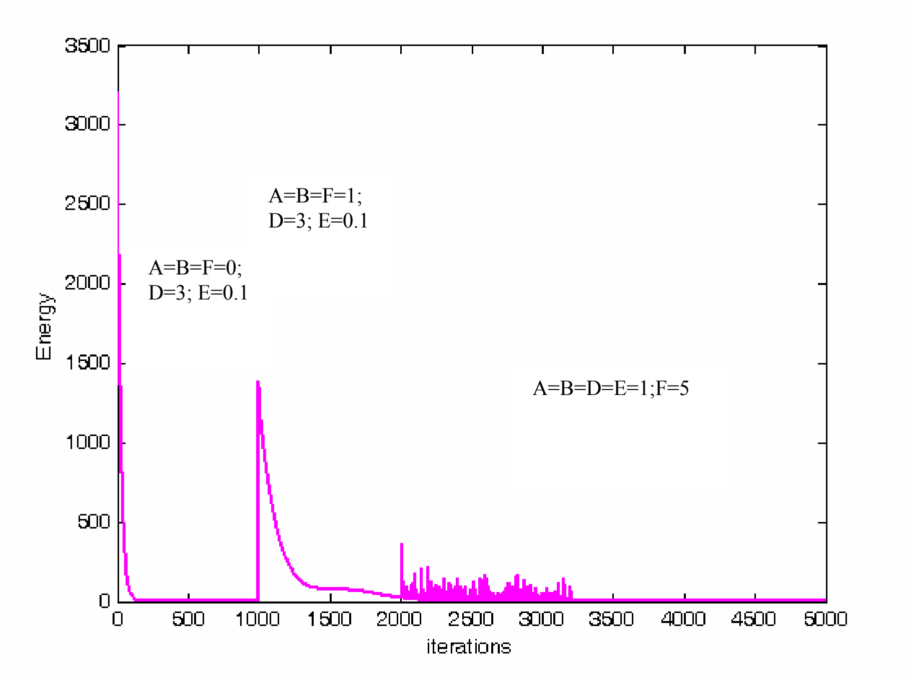


Figure 6.9 Energy evolution of the network during simulation

Table 6.3 Step size values in three phases of the simulation experiment

Iterations\Parameters	η_E	η_T	η_X	η_Z	η_Y	η_{YO}
1:1000	0.001	0.005	0.001	1	0.0005	1
1001:2000	0.001	0.005	0.001	0.001	0.05	0.000001
2001:5000	0.001	0.005	0.001	0.005	0.09	0.000001

Table 6.4 Slope values in three phases of the simulation experiment

Iterations\Parameters	λ_Y
1:1000	300
1001:2000	300
2001:5000	300+0.5×(iteration/1500)

From the simulation results $Z_{11}=Z_{23}=Z_{32}=Z_{43}=Z_{51}=Z_{61}=Z_{72}=Z_{82}=1$. According to the values of the assignment variables, job 1, job 5 and job 6 are performed on the first, job 3, job 7 and job 8 are performed on the second, and job 2, job 4 are performed on the third machine respectively. $Y_{023}=Y_{061}=Y_{072}=1$ implies that job 6 is the first job to be performed on the first machine, job 7 is the first job to be performed on the second machine, and job 2 is the first job to be performed on the third machine. Since the variables that determine the precedence relations between the jobs are $Y_{151}=Y_{611}=Y_{382}=Y_{732}=Y_{243}=1$, the first job precedes job 5 on machine 1, job 6 precedes job 1 on machine 1, job 3 precedes job 8 on machine 2, job 7 precedes job 3 on machine 2, and job 2 precedes job 4 on machine 3.

6.5 Conclusions

In this chapter, the problem of scheduling a set of independent jobs with sequence dependent setups and distinct due dates, on non-identical multi-machines to minimize the sum of weighted earliness and tardiness was studied. A dynamical coupled network composed of two maximum, three piecewise linear and one log-sigmoid sub-neural networks all of which interact with each other was proposed for the solution of the problem known to be NP-hard. By using the proposed interconnected network approach, the complexity of the considered problem which is expressed with too many constraints was reduced. While some of the constraints were eliminated from the energy function using maximum networks, some were satisfied using a log-sigmoid network. By this way, the effort of finding some penalty parameters was diminished, and by incorporating a kind of mean field annealing into the network, the performance of the network was improved. Implementation of the network in hardware will make significant progress in its computational efficiency.

Although a large body of literature exists for solving single machine scheduling problems involving earliness and tardiness penalties, there are few papers aim to minimize the sum of weighted earliness and tardiness and dealing with non-identical multi machine scheduling problems involving sequence dependent setup times and distinct due dates. To the best of our knowledge, there is no previously published article that tried to solve this NP-hard problem using neural networks. In addition, the application of competitive type networks to scheduling problems is scarce. So, we believe that this attempt to solve the problem considered will make a contribution to the relevant literature. Furthermore, the proposed Hopfield-like network uses time-varying penalty parameters to overcome the tradeoff problem, one of the important drawbacks of the penalty function approach. Obtaining an optimal solution using the proposed approach makes it attractive for applications of problems with larger size.

Future research should consider the hybridization of ANNs with other metaheuristics for the problems considered. In addition, extension of the results to large sized problems and implementation of parallel neural processing will also be worthwhile.

CHAPTER SEVEN
RESULTS ANALYSIS and DISCUSSION

In this thesis, we have applied Hopfield type neural networks to solve the problems considered. In general, we can summarize the entire procedure of employing Hopfield type dynamical neural networks to solve scheduling problems as follows.

Firstly the problem has to be modeled using linear or non linear programming techniques. Then, using the penalty function approach, the constrained problem is converted into an unconstrained problem. In other words, using this approach, the constraints of the problem are included in the energy function as penalty terms which can be considered as a Lagrangian relaxation of the constraints. Therefore, the energy function of the network will include the original objective function of the problem and the penalty terms corresponding to each constraint. To summarize the approach, let us consider the following minimization problem which includes both equality and inequality constraints.

$$
\begin{array}{ll}
\min & f(x,y) \\
a_i(x,y) \le 0 & i = 1,2,\ldots,n \\
b_j(x,y) = 0 & j = 1,2,\ldots,m \\
x_{ij} \in \{0,1\} & i = 1,\ldots,l;\ j = 1,\ldots k \\
y_j \in R & j = 1,2,\ldots,m
\end{array}
$$

The penalty terms,

$$A(x,y) = \sum_{i=1}^{m} v[(a_i(x,y)],$$

corresponding to the constraints represented by equalities, and the penalty terms,

$$B(x,y) = \sum_{j=1}^{m} b_j^{\,2}(x,y)$$

corresponding to the constraints represented by inequalities are obtained.

Here, v represents the penalty function and $\nu(\varepsilon) = \varepsilon^2$ for all ε > 0 and $\nu(\varepsilon) = 0$ for all ε ≤ 0 or $\nu(\varepsilon) = \varepsilon$ for all ε > 0 and $\nu(\varepsilon) = 0$ for all ε ≤ 0 can be used.

The penalty terms corresponding to the binary constraints

$$X(x) = \sum_{i=1}^{l} \sum_{j=1}^{k} x_{ij}(1 - x_{ij}),$$

and the energy function of the problem

$$E(x,y) = P1\, f(x,y) + P2 \sum_{i=1}^{m} \nu[(a_i(x,y)] + P3 \sum_{j=1}^{m} b_j^{\,2}(x,y) + P4 \sum_{i=1}^{l} \sum_{j=1}^{k} x_{ij}(1 - x_{ij})$$

can be written. In the energy function, *P1*, *P2*, *P3* and *P4* are penalty coefficients.

In addition to the objective function *f(x,y)*, the energy function *E(x,y)* also includes penalty terms, and when any constraint is violated, a cost value which causes a high energy appears. In a valid schedule, the penalty terms corresponding to constraints become zero. That is, the penalty coefficients will have no effect in the energy function.

After defining the energy function to be employed, the dynamics for the gradient network are obtained by gradient descent on the energy function.

$$\frac{dUx_{ij}}{dt} = -\frac{\partial E}{dVx_{ij}} \qquad i = 1,\dots,l; \quad j = 1,\dots,k$$

$$\frac{dUy_j}{dt} = -\frac{\partial E}{dVy_j} \qquad j = 1,\dots,m$$

The solution of equations of motion may be accomplished via the use of Euler's approximation. Then, neuron outputs are calculated by V=g (U), where g (.) is the activation function, *U* is the input and *V* is the output of the neuron. In words, an optimization problem is first mapped onto a neural network in such a way that the network configurations correspond to possible solutions to the problem. A function of neuronal states, called the energy function, is constructed. This function is proportional to the cost function of the problem. The dynamics of the network is determined so that the energy function (cost function) is minimized as the neurons update (Wang, 1997).

Generally, scheduling problems represented by mathematical models involve many constraints due to the complexity of the problem. The more constraints the model includes, the more difficult it becomes to solve the problem using neural networks, since a tradeoff occurs between the terms to be minimized. We propose below alternative ways to simplify the energy function and eliminate the effort of searching for the values of some of the penalty parameters.

- Since each job has to be processed on only one machine at a time, scheduling problems are generally subject to typical constraints such as,

$$\sum_{j=1}^{k} x_{ij} = 1 \qquad\qquad 1 \le i \le l$$

When a job is assigned to a machine, the assignment variable takes the value of 1, otherwise it becomes 0. For instance, if the schedule is represented by a 2-D $l \times k$ network, the binary output of the *ij*th neuron VX_{ij}=1 if a job is assigned in the *i*th row and the *j*th column, and VX_{ij}=0 otherwise. These binary constraints can be satisfied using a log-sigmoid activation function and by increasing the slope of the activation function when these constraints are violated.

The following maximum function satisfies both the assignment of one job to only one machine and also the binary constraint $x_{ij} \in \{0,1\}$. Here, the neuron outputs are represented by V's and the inputs by U's respectively.

$$Vx_{ij} = \begin{cases} 1 & if \quad Ux_{ij} = \max(Ux_{i1}, Ux_{i2}, ..., Ux_{ik}) \\ 0 & otherwise \end{cases}$$

- To represent real (continuous) valued variables linear activation functions can be used. If these variables must be greater than or equal to zero, then a piecewise linear function can be employed. By this way, there will be no need to consider the penalty term corresponding to this constraint. To represent binary (discrete) variables, a log-sigmoid function such as logsig(n) = 1 / (1 + exp(-n)), can be suggested. It should be noted that the convergence of the network, which means that the trajectories of the dynamical network end at one of the equilibria, should be proved before the use of dynamical gradient networks. However, although employing log-sigmoid functions do not cause convergence problems, in solving optimization problems using maximum functions and piecewise linear functions the convergence of the network should be analyzed carefully.

In the light of the procedure explained above, in this thesis, we have studied two different parallel machine scheduling problems. The first one was identical parallel machine scheduling problem with the objective of the makespan minimization, and the second one was the scheduling of a set of independent jobs with sequence-dependent setups and distinct due dates on non-identical multi-machines to minimize the total weighted earliness and tardiness. Both of the problems considered were NP-hard. Since the first problem includes fewer constraints than the second one, its energy function involved fewer penalty terms to enforce the constraints. Therefore, although a tradeoff exists between the penalty terms to be minimized in both of the problems, solving the first problem was easier than solving the second problem. For these problems, time dependent penalty parameters were used. For the solution of the first problem, the proposed gradient network included two types of neurons: a continuous type neuron to represent variable denoting makespan, and discrete types of neurons to represent binary valued assignment variables. Linear type activation function was used to model the relation between the output and the input of the continuous type neuron, and a log-sigmoid function was employed for the discrete type neuron.

From the simulation results, it was seen that the gradient network results were closer to the LPT results and in all the simulation runs for different initialization schemes, the network converged to feasible solutions. In other words, the convergence rate was 100 %. The performance of the network was also tested for large values of n (n = 500). The network produced better results for large values of n and small values of m, and only for n=500, it resulted one non-feasible solution in one of the 20 initialization schemes for one data set.

In the second application, inclusion of non-identical machines, sequence-dependent setup times, distinct due dates made the problem more complex, therefore many constraints were needed to represent the problem by using a mixed integer programming formulation. Due to the complicated structure of the problem, the energy function involved many penalty terms corresponding to each constraint of the problem. However, the quality of the final solution was very sensitive to the values of the penalty coefficients used and the effort of performing a search for the proper values for these coefficients was onerous (Van den Bout & Miller, 1988). Although some of the constraints such as assignment and binary constraints were tried to be satisfied by using maximum networks and log-sigmoid networks respectively, the problem still involved many constraints which resulted a greater tradeoff problem than the first problem. Another difficulty in solving this problem was that some variables were included in more than one constraint which made all constraints dependent to each other. Using a trial end error procedure, the proper values of the penalty parameters were found and an optimum solution which makes the approach attractive for solving larger instances was obtained for an example problem.

It should be emphasized that the choice of appropriate parameters always plays an important role in obtaining satisfactory solution quality. The performance of the neural network is very sensitive to penalty parameter values. If the parameters are chosen very small, an infeasible solution may be obtained. On the other hand, large values of the penalty parameters will ensure a feasible solution, but may create a poor local minimum solution. In order to satisfy each penalty term, its associated penalty parameter can be increased but this results an increase in other penalty terms and a tradeoff occurs. In our applications, the tradeoff problem was also one of the main problems we faced and we proposed to use time varying penalty parameters to solve this problem. By this method, we have obtained satisfactory solutions in the first application. However, the solutions obtained were generally local minimums instead of global optimum solutions. In addition to this, although the use of time varying penalty parameters during simulation experiments relieved some difficulty in solving the second problem, finding an optimal solution required too much trial and error efforts due to the complex nature of the problem.

Other parameters, the initial states of the network, the step sizes, and the slope of the activation functions used had also a great influence on the solutions of our problems and need to be carefully selected. In our first application, it is seen that even for the same datasets, different initial states of the gradient network generated different results some of which were better, some of which were worse than the LPT rule result and in the second application, it was very difficult to find the initial states that result a feasible solution. The step sizes and the slopes of the activation functions in both of the applications were determined by a trial and error process and in the second application, a kind of mean field annealing is incorporated into the network to slowly increase the slope of the log-sigmoid activation function used and by this way the slope of the activation function became time-varying and the binary constraints on the precedence units are satisfied.

In the following chapter, we summarize the entire thesis and point out possible directions for future research.

CHAPTER EIGHT
CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this concluding chapter, we summarize what has been accomplished in this thesis, and describe some potential future work to extend the present results for the discussed problems.

8.1 Conclusions

Scheduling refers to the assignment of limited resources over time to accomplish certain tasks. Since most of the scheduling problems are known to be NP-hard, approximate or heuristic approaches have been developed for solving these problems. During the last decades, many researchers have investigated the use of neural networks for solving scheduling problems.

In the thesis, first, an overview of scheduling together with the methods used in scheduling was given. A literature review of artificial neural network applications in production scheduling was provided after dynamical systems for combinatorial optimization were explained. Then, we developed dynamical neural networks to deal with two parallel machine scheduling problems. Our objective was two fold; to solve these problems using neural networks and to compare its performance with other techniques'. Although neural network approach has been admitted as a promising alternative to solving a variety of combinatorial optimization problems, few works relate neural network to applications of parallel machine scheduling problems. From the theoretical viewpoint, the parallel machine scheduling problem is a generalization of the single machine, and a special case of the flexible flow. It is also important from the practical point since the occurrence of resources in parallel is very common in real world (Pinedo, 1995). To the best of our knowledge, this study will be the first attempt to solve the considered NP hard problems using neural networks.

For the solution of the first problem, the identical parallel machine scheduling problem with the objective of minimizing makespan, we employed a dynamical gradient network. After the appropriate energy function was constructed by using a penalty function approach, the dynamics were defined by steepest gradient descent on the energy function. In order to overcome the tradeoff problem encountered in using the penalty function approach, a time varying penalty coefficient methodology was proposed to be used during simulation experiments. We analyzed the effect that the initial conditions of the network have on the performance on 5 different data sets by running each data set 20 times for different sizes of jobs and machines. The results of the simulation experiments indicated that applying the proposed method generated good results comparable with that of Longest Processing Time dispatching rule. The network converged to feasible solutions in all the simulation runs for different initialization schemes except in one of the 20 initialization schemes for one data set for n=500 and better results were obtained for large number of jobs and small number of machines.

The second problem was the scheduling of a set of independent jobs with sequence-dependent setups and distinct due dates on non-identical multi-machines to minimize the total weighted earliness and tardiness. In relevant literature, most of the researches on earliness and tardiness were conducted in a single machine setting. However, multi machines exist in many real world situations, and the regular performance measures, such as minimizing the total tardiness, number of tardy jobs, maximum tardiness and flowtime, can not meet the needs of many practical problems. The motivation behind this study is that although many researchers have focused on the parallel machine problems, there is a lack of research on the multi-machine scheduling problem with the non-regular performance measure of minimizing the earliness and tardiness. Firstly, the original mixed integer formulation of the problem was modified by adding one more constraint to the model to prevent the assignment of two jobs at the same time to the first position. Then, an interconnected neural network was developed to solve the problem. The proposed network is composed of two maximum networks, three piecewise linear networks and one log-sigmoid network all of which interact with each other.

The motivation for using maximum networks was to reduce the network complexity by incorporating competitive mechanism into the network and to obtain a simplified energy function. Additionally, the log-sigmoid network helps us to get rid of some of the binary constraints. Convergence of the network was analyzed by using the extension of the La Salle's invariance principle developed for the systems with discontinuous right hand sides. Again, in this application, a time varying penalty coefficient methodology was proposed to be used during simulation experiments to overcome the tradeoff problem encountered in using the penalty function approach. The proposed approach was tested on a scheduling problem. An optimal solution which may be promising for the applications of large size problems was obtained.

In general, we can say that the results obtained using the proposed neural network models were acceptable in terms of solution quality. However, with the implementation of parallel processing, full benefits of the neural network approach can be explored and assessed. The main benefit one can expect from using the neural networks in performing task optimization is the additional efficiency gained from implementation of parallel neural processing. Parallel processing and parallel computation has been well accepted as a legitimate and effective way for speed improvement in solving many combinatorial optimization problems (Censor & Zenios, 1997; Migdalas, Pardalos, & Story, 1997). However, a challenge with the parallel approaches is that many tasks cannot be easily or possibly broken down into a parallel structure so that the parallel processing can be performed. Because of the neural network's inherent parallel nature of processing units and network structure, once a problem is formulated into a neural network model, it will be in a ready mode to realize parallel processing. In other words, the neural network can be viewed as a natural vehicle to convert a problem into a parallel format. For full exploration of the neural network's potential in optimization, we need to firstly formulate a problem into a neural network model and then implement the neural network algorithms on a multiple processor machine or on a parallel-computing platform. Although we can find a quite number of studies that aim to implement parallel computation in using neural networks (Salleh & Zomaya, 1999; Saratchandran, Dundararajan, & Foo, 1996), most reported works in the literature have fulfilled only the first half of the process. Since the neural network computation in our experiments works also in serial mode, the experimental results reported do not reflect the potential of the true benefit of the neural network approach. With fast advance of high technology, parallel processing facilities will become inevitably more popular and easy to access. To this extent, we can expect a great improvement in computation time using the neural network approach (Hao, Lai, & Tan, 2004).

8.2 Future Research Directions

The followings which are possible extensions of this study are suggested for future research.

- One of the major shortcomings of the proposed networks was determination of the parameters required for the simulation of the proposed networks, such as the penalty parameters, the slope of the activation functions, the number of iterations and the step sizes, by trial and error. This is a tedious process, and the parameter values obtained might not be the optimal values for this study. The methodology for obtaining appropriate parameters for the development of a neural network that will yield more precise results should be considered in a future study.

- There exists a great potential for application of neural networks in conjunction with other optimization techniques. A number of different techniques and metaheuristics can be combined with neural networks to tackle the problems considered.

- Much work has to be done to compare the performance of neural network models with the performance of other existing methods.

- Hopfield like dynamical networks were chosen to map our problems since they facilitated easy mapping of the problem. Likewise, these networks can be implemented in hardware for making them attractive alternatives to classical heuristic approaches. It is obvious that if the proposed networks can be implemented in hardware, there will be a tremendous reduction in computational times.

- Dynamical gradient networks suffer from serious problems of getting stuck at local minima, having high sensitivity to parametric changes and tradeoff problem among these parameters. Several problems still exist although a penalty function approach whose coefficients vary with time were used to alleviate the tradeoff problem in this study. To overcome the local minima, stochastic methods such as simulated annealing can be integrated with ANNs. By introducing a probability for the acceptance of a new state, the network occasionally accepts transitions to states with higher energy and thus can escape from local minima. Replacing sigmoidal activation function with a stochastic decision type activation function, adding noise to the weights of the network or to the biases of the network are some of the main methods used to embed stochasticity into the Hopfield network (Smith, 1999).

- One of the other issues for future research may be to introduce evolution to adjust the topology and the parameters of ANNs automatically or to search for the ways of developing automatic parameter controlling methods to overcome the need of tuning the parameters by a trial and error.

- The performance of the proposed networks depends on the choice of the initial states. Another area on which future research has to focus may be to propose new neural network models that are less sensitive to the initial states.

- Other scheduling objectives, setup times, the mean flow times or the weighted number of tardy jobs minimization, in parallel machine scheduling can be studied. Especially for the first application, non-identical parallel machine case is worthy of exploration.

- It may also be interesting to evaluate the proposed networks' performance for real time scheduling systems.

REFERENCES

Aarts, E., & Korst, J. (1989). *Simulated annealing and Boltzmann machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Chichester: John Wiley & Sons.

Aarts, E., & Lenstra, J.K. (Eds.). (1997). Local Search in Combinatorial Optimization. Chichester: John Wiley & Sons.

Abdul-Razaq, T., & Potts, C. (1988). Dynamic programming state-space relaxation for single-machine scheduling. *Journal of the Operational Research Society*, *39*, 141–152.

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, *34* (3), 391- 401.

Agarwal, A., Pirkul, H., & Jacob, V.S. (2003). Augmented neural networks for task scheduling. *European Journal of Operational Research*, *151* (3), 481-502.

Aggarwal, J.K., & Vidyasagar, M. (1977). Nonlinear systems: stability analysis, Dowden, Hutchinson and Ross, Inc. Stroudsburg, Pennsylvannia.

Ahmed, I., Kwok, Y.-K., Ahmad, I., & Dhodhi, M. (2001). Scheduling parallel programs using genetic algorithms. In A. Y. Zomaya, F. Ercal, and S. Olariu, editors. Solutions to Parallel and Distributed Computing Problems, 231-254. New York: John Wiley and Sons.

Aiyer, S.V.B., Niranjan, M. & Fallside, F. (1990). A theoretical investigation into the performance of the Hopfield model. *IEEE Transactions on Neural Networks, 1*, 204–215.

Akiyama, Y., Yamashita, A., Kajiura, M., & Aiso, H. (1989). Combinatorial Optimization with Gaussian Machines. *Proceedings IEEE International Joint Conference on Neural Networks*, *1*, 533–540.

Akyol, D.E. (2004). Application of neural networks to heuristic scheduling algorithms. *Computers & Industrial Engineering, 46,* 679-696.

Akyol, D.E., & Bayhan, G.M. (2005). A Coupled Gradient Network Approach for the Multi Machine Earliness and Tardiness Scheduling Problem. In Gervasi, O., Gavrilova, M.L., Kumar, V., Lagana, A., Lee, H.P., Mun, Y., Taniar, D., & Tan C.J.K. (Eds.). *Lecture Notes in Computer Science, LNCS 3483* (596-605). Berlin: Springer.

Akyol, D.E., & Bayhan, G.M. (2006). A Review on Evolution of Scheduling with Neural Networks. *Computers & Industrial Engineering.* (under review)

Allahverdi, A., & Al-Anzi, F.S. (2005). A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application. *Computers & Operations Research*, *33* (4), 1056-1080.

Alvarez, A. (2002). A neural network with evolutionary neurons. *Neural Processing Letters, 16,* 43–52.

Arizono, I., Yamamoto, A., & Ohta, H. (1992). Scheduling for minimizing total actual flow time by neural networks. *International Journal of Production Research*, *30* (3), 503-511.

Arkin, E., & Roundy, R.O. (1991). Weighted-tardiness scheduling on parallel machines with proportional weights. *Operations Research*, 39, 64-81.

Artiba, A., & Elmaghraby, S.E. (1997). *The Planning and Scheduling of Production Systems: Methodologies and applications*. London: Chapman & Hall.

Arzi, Y., & Iaroslavitz, L. (1999). Neural network-based adaptive production control system for a flexible manufacturing cell under a random environment. *IIE Transactions, 31*, 217-230.

Askin, R., & Standridge, C. (1993). *Modeling and analysis of manufacturing systems*. New York: John Wiley & Sons.

Bagchi, U., Sullivan, R.S., & Chang, Y.L. (1986). Minimizing mean absolute deviation of completion times about a common due date. *Naval Research Logistics Quarterly, 33*, 227–240.

Baker, K.R. (1974). *Introduction to Sequencing and Scheduling*. New York: Wiley.

Baker, K.R., & Scudder, G.D. (1990). Sequencing with earliness and tardiness penalties: a review. *Operations Research, 38*, 22-36.

Balakrishan, N., Kanet, J.J., & Sridharan, S.V. (1999). Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computers & Operations Research*, *26*, 127-141.

Baxter, J. (1992). The evolution of learning algorithms for artificial neural networks. *Complex Systems.* In D. Green, & T. Bossomaier. (Eds.). (313-326). IOS Press.

Belegundu, A.D., & Chandrupatla, T.R. (1999). *Optimization Concepts and Applications in Engineering.* New Jersey: Prentice-Hall

Belew, R.K., & Booker L.B. (Eds.) (1991). *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Mateo, California: Morgan Kaufmann

Bellman, R. (1957). *Dynamic Programming*. Princeton, N.J.: Princeton University Press.

Ben-Daya, M., & Al-Fawzan, M. (1998). A tabu search approach for the flow shop scheduling problem. *European Journal of Operational Research*, *109*, 88-95.

Blackstone, J.H., Philips, D.T. & Hogg, G.L. (1982). The state of the art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, *20* (1), 27-45.

Blazewicz, J., Dror, M., & Weglarz, J. (1991). Mathematical programming formulations for machine scheduling: a survey. *European Journal of Operational Research*, *51,* 283–300.

Bornholdt, S., & Graudenz, D. (1992). General asymmetric neural networks and structure design by genetic algorithms. *Neural Networks, 5* (2), 327–334.

Brandt, R.D., Wang, Y., Laub, A.J., & Mitra, S.K. (1988). Alternative Networks for Solving the Travelling Salesman Problem and the List-Matching Problem. *Proceedings of the International Conference on Neural Networks*, *2,* 333-340.

Brucker, P. (2003). Scheduling Algorithms. New York: Springer Verlag.

Cagnina, L., Esquivel, S., & Gallard, R. (2004). Particle swarm optimization for sequencing problems: a case study. *Proceedings of the Congress on Evolutionary Computation*, *1,* 536–541.

Cakar T., & Cil, I. (2004). Artificial neural networks for design of manufacturing systems and selection of priority rules. *International Journal of Computer Integrated Manufacturing, 17* (3), 195-211.

Cakir, Y. (2002). Dynamical Solvers for Linear and Quadratic Optimization. Ph.D. Thesis, Istanbul Technical University.

Carpenter, G. A., & Grossberg, S. (1987). ART: 2 self-organization of stable category recognition codes for analog input patterns. *Applied Optics, 26*, 4919-4930.

Censor, Y., & Zenios, S.A. (1997). *Parallel Optimization: Theory, Algorithms and Applications*. New York: Oxford University Press.

Cha, Y., & Jung, M. (2003). Satisfaction assessment of multi-objective schedules using neural fuzzy methodology. *International Journal of Production Research, 41* (8), 1831-1849.

Chalmers, D.J., 1990. The evolution of learning: An experiment in genetic connectionism. *Proceedings of the 1990 Connectionist Summer School Workshop*, 81–90.

Chen, B., Potts, C.N., & Woeginger, G.J. (1998). A review of machine scheduling: complexity, algorithms and approximation. In D.-Z. Du, & P. M. Pardalos, (Eds.). Handbook of Combinatorial Optimization (21-169). Dordrecht: Kluwer Academic Publishers.

Chen, F.F., Huang, J., & Centeno, M.A. (1999). Intelligent scheduling and control of rail-guided vehicles and load/unload operations in a flexible manufacturing system. *Journal of Intelligent Manufacturing, 10,* 405-421.

Chen, M., & Y. Dong, Y. (1999). Applications of neural networks to solving SMT scheduling problems-a case study. *International Journal of Production Research, 37,* 4007-4020.

Chen, R.M, & Huang, Y.M. (2001). Competitive neural network to solve scheduling problems. *Neurocomputing, 37,* 177-196.

Chen, R.M., & Huang, Y.M. (2001). Multiprocessor Task Assignment with Fuzzy Hopfield Neural Network Clustering Technique. *Neural Computing & Applications, 10,* 12-21.

Chen, W., & Muraki, M. (1997). An action strategy generation framework for an on-line scheduling and control system in batch processes with neural networks. *International Journal of Production Research, 35* (12), 3483-3507.

Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms: Part I. Representation. *Computers and Industrial Engineering, 30,* 983-997.

Cheng, R., Gen, M., & Tsujimura, Y. (1999). A tutorial survey of job-shop scheduling problems using genetic algorithms: Part II. Hybrid Genetic Search. *Computers and Industrial Engineering, 36,* 343-364.

Cheng, T., & Sin, C. (1990). A State-of-the-Art Review of Parallel-Machine Scheduling Research. *European Journal of Operation Research, 47*, 271-292.

Cheung, J.Y. (1994). Scheduling. In C. Dagli, (Ed.). Artificial Neural Networks for Intelligent Manufacturing (159-193). London: Chapman & Hall.

Cho, S.B., & Shimohara, K. (1998). Evolutionary Learning of Modular Neural Networks with Genetic Programming. *Applied Intelligence, 9*, 191-200.

Chryssolouris, G., Lee, M., & Domroese, M. (1991). The use of neural networks in determining operatioal policies for manufacturing systems. *Journal of Manufacturing Systems, 10* (2), 166-175.

Chung, P.C., Tsai, C.T., Chen, E.L., & Sun, Y.N. (1994). Polygonal approximation using a competitive Hopfield neural network. *Pattern Recognition, 27* (11), 1505–1512.

Coello Coello, C. A., & Lechuga, M. S. (2002). MOPSO: a proposal for multiple objective particle swarm optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, 1051-1056.

Coffman, E.G., Garey, M.R., & Johnson, D.S. (1978). An application of bin-packing to multi-processor scheduling. *SIAM Journal of Computing*, *7*, 1-17.

Cortez, P., Rocha, M., & Neves, J. (2002). A Lamarckian approach for neural network training. *Neural Processing Letters, 15*, 105-116.

Crosher, D. (1993). The artificial evolution of a generalized class of adaptive processes. *Proceedings of AI'93 Workshop on Evolutionary Computation, 18-36.*

Dagli, C., & Sittisathanchai, S. (1993). Genetic neuro-scheduler for job shop scheduling. *Computers and Industrial Engineering, 25* (1-4), 267-270.

De, P., Ghosh, J.B, & Wells, C.E. (1994). Due dates and early/tardy scheduling on identical parallel machines. Naval Research Logistics, *41*, 17-32.

Deoser, C.A. (1970). *Notes for a second course on linear systems*. Berkeley: Van Nostraund Reinhold Company.

Ding, F.Y., & Kittichartphayak, D. (1994). Heuristics for scheduling flexible flow lines. *Computers and Industrial Engineering, 26*, 27-34.

Dogan, H. (2004). Gradient networks design for clustering in novel optimization frameworks. Ph.D. Thesis, Dokuz Eylul University.

Dogan, H., & Guzelis, C. (2006) Robust and Fuzzy Spherical Clustering by a Penalty Parameter Approach. *IEEE Transactions on Circuits and Systems-II, 53*(8), 637-641.

Eglese, R.W. (1990). Simulated annealing: A tool for operational research. *European Journal of Operational Research*, *46*, 271-281.

Eiselt, H.A., & Sandblom, C.-L. (2004). *Decision Analysis, Location Models, and Scheduling Problems.* Berlin Heidelberg: Springer-Verlag.

El-Bouri, A., Balakrishnan, S., & Popplewell, N. (2000). Sequencing jobs on a single machine: A neural network approach. *European Journal of Operational Research, 126*, 474-490.

El-Gallad, A. I., El-Hawary, M. E., & Sallam, A. A. (2001). Swarming of intelligent particles for solving the nonlinear constrained optimization problem. *Engineering Intelligent Systems for Electrical Engineering and Communications, 9*, 155-163.

Fang, J., & Xi, Y. (1997). Neural network design based on evolutionary programming. *Artificial intelligence in Engineering, 11*, 155-161.

Fang, L., & Li, T. (1990). Design of competition based neural networks for combinatorial optimization. *International Journal of Neural Systems*, *1*(3), 221-235.

Feng, S., Li, L., Cen, L., & Huang, J. (2003). Using MLP networks to design a production scheduling system. *Computers and Operations Research, 30,* 821–832.

Fogel, D.B., Fogel, L.J., & Porto, V.W. (1990). Evolving neural networks. *Biological Cybernetics, 63* (6), 487-493.

Fogel, L.J., Owens, A.J., & Walsh, M.J. (1966). *Artificial Intelligence Through Simulated Evolution*. New York: John Wiley & Sons.

Fonseca, D.J., & Navaresse, D. (2002). Artificial neural networks for job shop simulation. *Advanced Engineering Informatics, 16*, 241-246.

Fontanari J.F., & Meir, R. (1991). Evolving a learning algorithm for the binary perceptron. *Network: Computation in Neural Systems*, *2* (4), 353–359.

Foo, S.Y., & Takefuji, Y., & Szu, H. (1995). Scaling properties of neural networks for job-shop scheduling. *Neurocomputing*, *8*, 79–91.

Foo, Y.P.S., & Takefuji, Y. (1988a). Stochastic neural networks for solving job-shop scheduling: Part 1, problem presentation. *Proceedings of Joint International Conference on Neural Networks, 2,* 275-282.

Foo, Y.P.S., & Takefuji, Y. (1988b). Stochastic neural networks for solving job-shop scheduling: Part 2, architecture and simulations. *Proceedings of Joint International Conference on Neural Networks*, *2*, 283–290.

Foo, Y.P.S., & Takefuji,Y. (1988c). Integer linear programming neural networks for job-shop scheduling. *Proceedings of Joint International Conference on Neural Networks*, *2*, 341–348.

Forrest, S. (Ed.) (1993). *Proceedings of the Fifth International Conference on Genetic Algorithms*. California: Morgan Kaufmann Publishers.

Frangioni, A., Scutella, M.G., & Necciari, E. (1999). Multi-exchange algorithms for the minimum makespan machine scheduling problem. *Technical Report: TR-99-22*.

Friesen, D.K., & Langston, M.A. (1986). Evaluation of a MULTIFIT based scheduling algorithm, *J. Algorithm*, *7*, 35-59.

Friesen, D.K. (1987). Tighter bounds for LPT scheduling on uniform processors. *SIAM J. Computing, 16*, 554-560.

Fukushima, K. (1975). Cognitrion: A self-organizing multilayered neural network. *Biological Cybernetics, 20*, 121-136.

Funabiki, N., Takenaka, Y., & Nishikawa, S. (1997). A maximum neural network approach for N-queens problem. *Biol. Cybern*., *76*, 251–255.

Galan-Marin, G., & Munoz-Perez, J. (2001). Design and Analysis of Maximum Hopfield Networks. *IEEE Transactions on Neural Networks, 12* (2), 329-339.

Galan-Marin, G., Merida-Casermeiro, E., & Munoz-Perez, J. (2003). Modelling Competitive Hopfield Networks for the maximum clique problem. *Computers & Operations Research*, *30*, 603-624.

Gao, W. (2003). Study on new evolutionary neural network. *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, 1287-1292.

Garey, M.R., & Johnson, D.S. (1979). *Computer and intractability: a guide to the theory of NP completeness*. San Francisco: W.H Freeman.

Garey, M.R., Tarjan, R.E., & Wilfong, G.T. (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research, 13,* 330-348.

Geneste, L., & Grabot, B. (1997). Implicit versus explicit knowledge representation in a job shop scheduling decision support system. *International Journal of Expert Systems*, *10* (1), 37-52.

Glover F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research, 13,* 533-549.

Glover, F. (1989). Tabu Search-Part-I. *ORSA J. Comput*., *1,* 190-206.

Glover, F. (1990). Tabu Search-Part-II. *ORSA J. Comput*., *2,* 4-32.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. MA: Addison-Wesley.

Gomory, R.E.(1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society, 64,* 275-278.

Gomory, R.E. (1963). An algorithm for integer solutions to linear programs. In R. L. Graves & P. Wolfe, (Eds.). *Recent advances in Mathematical Programming* (269-302). New York: McGrawHill.

Graham, R.L. (1969). Bounds on multiprocessor timing anomalies. *SIAM Journal of Applied Mathematics*, *17,* 416-429.

Graham, R.L., Lawler, E.L., Lenstra, J.K., & Rinnooy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, *5,* 287-326.

Greenwood, G.W. (1997). Training partially recurrent neural networks using evolutionary strategies. *IEEE Transactions on Speech & Audio Processing, 5* (2), 192–194.

Grossberg, S. (1972). Neural expectation: Cerebellar and retinal analogs of cells fired by learnable or unlearned pattern classes. *Kybernetik, 10,* 49-57.

Grossberg, S. (1976a). Adaptive pattern classification and universal recording: I. Parallel development and coding of neural detectors. *Biological Cybernetics*, *23,* 121-134.

Grossberg, S. (1976b). Adaptive pattern classification and universal recording: II. Feedback, expectation, olfaction, illusions. *Biological Cybernetics, 23,* 187-202.

Hall, N.G. (1986). Single and multiple-processor models for minimizing completion time variance. *Naval Research Logistics Quarterly, 33*, 49-54.

Hamad, A., Sanugi, B., & Salleh, S. (2003). A neural network model for the common due date job scheduling on unrelated parallel machines. *International Journal of Computer Mathematics*, *80* (7), 845-851.

Hao, G., Lai, K.K., & Tan, M. (2004). A Neural Network Application in Personnel Scheduling. *Annals of Operations Research*, *128,* 65-90.

Hax, A., & Candea, D. (1984). *Production and Inventory Management.* Englewood Cliffs, NJ : Prentice Hall.

Haykin, S. (1994). *Neural Networks: A comprehensive foundation*. New Jersey: Prentice-Hall.

Heady, R., Zhu, Z. (1998). Minimizing the Sum of Job Earliness and Tardiness in a Multimachine System. *International Journal of Production Research*, *36,* 1619-1632.

Hedge, S., Sweet, J., & Levy, W. (1988). Determination of parameters in a Hopfield/Tank computational network. *Proceedings of the IEEE International Conference on Neural Networks*, *2,* 291-298.

Hinton, G.E., & Sejnowski, T.J. (1986). Learning and relearning in Boltzmann machines. In: D.E. Rumelhart, & J.L. McClelland, *Parallel Distributed Processing: Explorations in Microstructure of Cognition*. Cambridge: MIT Press.

Hirsch, M. W., & Smale, S. (1974). Differential equations, dynamical systems and linear algebra. Academic Press.

Hochbaum, D.S., & Shmoys, D. B. (1987). Using dual approximation algorithms for scheduling problems: Practical and theoretical results. *Journal of the Association for Computing Machinery, 34*, 144-162.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. Michigan: The University of Michigan Press.

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems*. Cambridge: The MIT Press.

Hong, Y.S., Lee, H., & Tahk, M.J. (2003). Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. *Engineering Optimization, 35* (1), 91-102.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of National Academy of Sciences, 79,* 2554-2558.

Hopfield, J. (1984). Neurons with graded response have collective computational properties like of two-state neurons. *Proceedings of the National Academy of Sciences of the USA, 81,* 3088-3092.

Hopfield, J., & Tank, T.W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics*, *52*, 141–152.

Hou, E., Ansari, N., & Ren, H. (1994). A genetic algorithm for multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed systems*, *5* (2); 113-120.

Hu, X., & Eberhart, R. C. (2002). Solving constrained nonlinear optimization problems with particle swarm optimization. *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics*, 203-206.

Huang, Y.M., & Chen, R.M. (1999). Scheduling multiprocessor job with resource and timing constraints using neural networks. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, 29* (4), 490-502.

Hübscher, R., & Glover, F. (1994). Applying Tabu Search with influential diversification to multiprocessor scheduling. *Computers and Operations Research*, *8*, 877-884.

Iazewicz, J.B., Domschke, W., & Pesch, E. (1996). The job shop scheduling problem: conventional and new solution techniques. *European Journal of Operational Research*, *93*, 1-33.

Igel, C., & Kreutz, M. (2003). Operator adaptation in evolutionary computation and its application to structure optimization of neural networks. *Neurocomputing*, *55*, 347-361.

Ilonen, J., Kamarainen, J.K., & Lampinen, J. (2003). Diffential Evolution Training Algorithm for Feed-Forward Neural Networks. *Neural Processing Letters, 17*, 93-105.

Islam, Md.M., Akita, H., Shahjahan, Md., & Murase, K. (2000). Representative Evolution: A simple and efficient algorithm for artificial neural network evolution. *Proceedings of the International Joint Conference on Neural Networks,* 585-590.

Jain, A.S., & Meeran, S. (1998). Job shop scheduling using neural networks. *International Journal of Production Research, 36* (5), 1249-1272.

Jeng, M.D., & Chang, C.Y. (1997). Non-energy based neural networks for job shop scheduling. *Electronics Letters, 33* (5), 399-400.

Jong, K.A.D. (1975). *An analysis of the behavior of a class genetic adaptive systems*. PhD thesis, Ann Arbor : University of Michigan.

Jozefowska, J., Milka, M., Rozycki, R., Waligora, G., & Weglarz, J. (1998). Local search metaheuristics for discrete-continuous problems. *European Journal of Operational Research, 107,* 354-370.

Jünger, M., Reinelt, G., & Thienel, S. (1995). Practical problem solving with cutting plane algorithms in combinatorial optimization. *Combinatorial Optimization: DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* 111-152.

Kaikhah, K., & Garlick, R. (2000). Variable hidden layer sizing in Elman recurrent neuro-evolution. *Applied Intelligence, 12,* 193-205.

Kamgar-Parsi, B., & Kamgar-Parsi, B. (1992). Dynamical Stability and Parameter Selection in Neural Optimization. *Proceedings of International Joint Conference on Neural Networks, 4*, 566-571.

Kanet, J.J. (1981). Minimizing the Average Deviation of Job Completion Times about a Common Due Date. *Naval Research Logistics Quarterly, 28*, 643-651.

Kenneddy, J., & Eberhart, R.C. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks, 4*, 1942-1948.

Kido, T., Takagi, K., & Nakanishi, M. (1994). Analysis and comparisons of genetic algorithm, simulated annealing, tabu search and evolutionary combination algorithm. *Informatica, 18,* 399-410.

Kim, C.O., Min, H.S., & Yih, Y. (1998). Integration of inductive learning and neural networks for multi-objective FMS scheduling. *International Journal of Production Research, 36* (9), 2497-2509.

Kim, H.B., Jung, S.H., Kim, T.G., & Park, K.H. (1996). Fast learning method for backpropagation neural network by evolutionary adaptation of learning rates. *Neurocomputing, 11* (1), 101-106.

Kim, S., Lee, Y.H., & Agnihotri, D. (1995). A hybrid approach to sequencing jobs using heuristic rules and neural network. *Production Planning and Control, 6*, 445-454.

Kim. D., Ahn, S., & Kang, D.S. (2000). Co-adaptation of self-organizing maps by evolution and learning. *Neurocomputing, 30*, 249-272.

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by Simulated Annealing. *Science, 220,* 671-680.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43*, 59-69.

Köppen, M., Teunis, M., & Nickolay, B. (1997). A neural network that uses evolutionary learning. *Proceedings of the 1997 IEEE Conference on Evolutionary Computation*, 635-639.

Koza, J.R., & Rice, J.P. (1991). Genetic generation of both weights and architecture for a neural network. *Proceedings of the International Joint Conference on Neural Networks, 2*, 397-404.

Lai, W.K., & Coghill, G.G. (1992). Genetic Breeding of Control Parameters for the Hopfield/Tank Neural Net. *Proceedings of the International Joint Conference on Neural Networks, 4*, 618-623.

Lee, H.C., & Dagli, C.H. (1997). A parallel genetic neuro scheduler for job shop scheduling problems. *International Journal of Production Economics, 51*, 115-122.

Lee, I., & Shaw, M.J. (2000). A neural-net approach to real time flow-shop sequencing. *Computers and Industrial Engineering, 38*, 125-147.

Lee, K.C., & Takefuji, Y. (1992). A generalized maximum neural network for the module orientation problem. *International Journal of Electronics, 72* (3), 331–355.

Lee, K.C., Funabiki, N., & Takefuji, Y. (1992). A parallel improvement algorithm for the bipartite subgraph problem. *IEEE Trans. Neural Networks, 3*, 139–145.

Lee, Y.H., Bhaskaran, K., & Pinedo, M. (1997). A heuristic to minimize the total weighted tardiness with sequence dependent setups. *IIE Transactions, 29*, 45-52.

Leung, J.Y.-T. (1989). Bin packing with restricted piece sizes. *Information Processing Letters, 31*(3), 145-149.

Li, D.C., Chen, L.S., & Lin, Y.S. (2003). Using Functional Virtual Population as assistance to learn scheduling knowledge in dynamic manufacturing environments. *International Journal of Production Research, 41* (17), 4011-4024.

Li, D.C., Wu, C., & Torng, K.Y. (1997). Using an unsupervised neural network and decision tree as knowledge acquisition tools for FMS scheduling. *International Journal of Systems Science, 28* (10), 977-985.

Li, H., Li, Z., Li, L.X., & Hu, B. (2000). A production rescheduling expert simulation system. *European Journal of Operational Research, 124*, 283-293.

Li, Z. (1996). Improving convergence and solution quality of Hopfield-type neural networks with augmented Lagrange multipliers. *IEEE Transactions on Neural Networks*, 7, 1507-1516.

Liansheng, G., Gang, S., Shuchun, W. (2000). Intelligent scheduling model and algorithm for manufacturing. *Production Planning and Control, 11*, 234-243.

Liebowitz, J., Krishnamurthy, V., Rodens, I., Houston, C., Liebowitz, A., & Zeide, J. (1997). Intelligent scheduling with GUESS: development and testing results. *Expert Systems, 14,* 119-128.

Liebowitz, J., Rodens, I., Zeide, J., & Suen, C. (2000). Developing a neural network approach for intelligent scheduling in GUESS. *Expert Systems, 17* (4), 185-190.

Liu, Y., & Yao, X. (2001). Evolving neural networks for Hang Seng stock index forecast. *Proceedings of the 2001 IEEE Congress on Evolutionary Computation, 256-260.*

Lo, Z.P., & Bavarian, B. (1993). Multiple job scheduling with artificial neural Networks. *Computers and Electrical Engineering, 19*, 87-101.

Looi, C. (1992). Neural network methods in combinatorial optimization. *Computers and Operations Research, 19* (3/4), 818-823.

Lu, W.Z., Fan, H.Y., & Lo, S.M. (2003). Application of evolutionary neural network method in predicting pollutant levels in downtown area of Hong Kong. *Neurocomputing, 51*, 387-400.

Luh, P. B., Zhao, X., & Wang, Y. (2000). Lagrangian Relaxation Neural Networks for Job Shop Scheduling. *IEEE Transactions on Robotics and Automation, 16*, 78-88.

Macleod, C., & Maxwell, G.M. (2001). Incremental Evolution in ANNs: Neural Nets which Grow. *Artificial Intelligence Review, 16*, 201-224.

Madureira, A.M. (1999). Meta-heuristics for the Single-Machine Scheduling Total Weighted Tardiness Problem. *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, 405-411.

Mandischer, M. (1995). Evolving recurrent neural networks with nonbinary encoding. *Proceedings of the International Conference on Evolutionary Computation*, 584-589.

Mandischer, M. (2002). A comparison of evolution strategies and backpropagation for neural network training. *Neurocomputing, 42*, 87-117.

McCulloch, W.S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*, 115-133.

McMullen, P.R. (2001). A Kohonen self organizing map approach to addressing a multiple objective, mixed model JIT sequencing problem. *International Journal of Production Economics, 72*, 59-71.

Merelo, J.J., Pat´on, M., Ca˜nas, A., Prieto, A., & Mor´an, F. (1993). Optimization of a competitive learning neural network by genetic algorithms. *Proceedings of the International Workshop on Artificial Neural Networks, Lecture Notes in Computer Science*, *686*, 185-192.

Migdalas, A., Pardalos, P.M., & Story, S. (Eds.). (1997). *Parallel Computation in Optimization*. New York: Kluwer Academic.

Min, H.S., & Yih, Y. (2003). Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system. *International Journal of Production Research, 41* (16), 3921-3941.

Min, H.S., Yih, Y., & Kim, C.O. (1998). A competitive neural network approach to multi-objective FMS scheduling. *International Journal of Production Research, 36* (7), 1749-1765.

Mokotoff, E. (2001). Paralel Machine Scheduling Problems: A Survey. *Asia-Pacific Journal of Operational Research*, *18,* 193-242.

Moriarty, D.E., & Miikkulainen, R. (1996). Efficient Reinforcement Learning through Symbiotic Evolution. *Machine learning, 22,* 11-32.

Mostaghim, S., & Teich, J. R. (2003). Strategies for finding local guides in multi-objective particle swarm optimization. *Proceedings of the IEEE Swarm Intelligence Symposium*, 26-33.

Murtadi, A. M. (2001). *Scheduling of jobs on parallel machines using genetic algorithms*. Ph.D. Dissertation, University of Windsor, Canada.

Mühlenbein, H., Schomisch, M., & Born, J. (1991). The parallel genetic algorithm as function optimizer. *Parallel Computing, 17,* 619-632.

Nawaz, M., Enscore, E., & Ham, I. (1983). A heuristic algorithm for the n-job, m-machine flowshop sequencing problem. *Omega, 11*, 91-95.

Nemhauser, G.L., & Wolsey, L.A. (1988). *Integer and Combinatorial Optimization,* New York: Wiley.

Osman, I.H. (2002). Preface, Focused issue on applied meta-heuristics. *Computers and Industrial Engineering*, 205-207.

Osman, I.H., & Potts, C.N. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega*, 17, 551-557.

Ovacık, I., & Uzsoy, R. (1997). *Decomposition Methods in Large Scale Job Shops*. Boston: Wiley and Sons.

Ow, P.S, & Morton, T.E. (1988). Filtered beam search in scheduling. *International Journal of Production Research, 26,* 35–62.

Ow, P.S, & Morton, T.E. (1989). The single machine early/tardy problem. *Management Science, 35,* 177–191.

Palmes, P.P., Hayasaka, T., & Usui, S. (2003). Evolution and Adaptation of Neural Networks. *Proceedings of the International Joint Conference on Neural Networks*, *1*, 478-483.

Panwalkar, S.S., & Iskander, W. (1977). A Survey of Scheduling Rules. *Operations Research*, *25* (1), 45-61.

Papadimitriou, C.H., & Steiglitz, K. (1998). Combinatorial Optimization: Algorithms and Complexity. New York: Dover Publications.

Pardalos, P. M., & Resende, M.G. (Eds.). (2002). *Handbook of Applied Optimization*. Oxford University Press.

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

Parker, D.B. (1985). *Learning logic: Casting the cortex of the human brain in silicon. Technical Report*, TR-47. Center for Computational Research in Economics and Management Science, Cambridge, MA: MIT Press.

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization method for constrained optimization problems. *Proceedings of the Euro International Symposium on Computational Intelligence*. In P. Sincak, J. Vascak, V. Kvasnicka, & J. Pospichal, (Eds.). *Intelligent Technologies - Theory and Applications: New Trends in Intelligent Technologies.* (214-220). IOS Press.

Patel, D. (1996). Using genetic algorithms to construct a network for financial prediction. *Proceedings of SPIE: Applications of Artificial Neural Networks in Image Processing*, 204-213.

Peterson, C., & Anderson, J.R. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems, 1*, 995-1019.

Philipoom, P.R., Rees, L.R., & Wiegmann, L. (1994). Using neural networks to determine internally-set due date assignments for shop scheduling. *Decision Sciences, 25* (5/6), 825-851.

Pinedo, M. (1995). *Scheduling Theory, Algorithms, and Systems*. New Jersey, USA: Prentice Hall.

Pinedo, M., & Singer, M. (1997). A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. *Naval Research Logistics, 46*, 1-17.

Pinto, J.M., & Grossmann, I.E. (1998). Assignment and Sequencing Models for the Scheduling of Chemical Processes. *Annals of Operations Research, 81*, 433-466.

Porto, V.W., Fogel, D.B., & Fogel, L.J. (1995). Alternative neural network training methods. *IEEE Expert: Intelligent Systems and Their Applications, 10*, 16-22.

Potvin, J.Y., & Smith, K.A. (2003). Artificial Neural Networks for Combinatorial Optimization. In: F. Glover, & G. Kochenberger, *Handbook of Metaheuristics* (429-455). Boston: Kluwer Academic Publishers.

Priore, P., Fuente, D., Pino, R., & Puente, J. (2003). Dynamic scheduling of flexible manufacturing systems using neural networks and inductive learning. *Integrated Manufacturing Systems, 14* (2), 160-168.

Pujol, J.C.F., & Poli, R. (1998). Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence, 8,* 73-84.

Raaymakers, W.H.M., & Weijters, A.J. M. M. (2003). Makespan estimation in batch process industries: A comparison between regression analysis and neural networks. *European Journal of Operational Research, 145*, 14-30.

Rabelo, L., & Alptekin, S. (1990). Adaptive scheduling and control using artificial neural networks and expert systems for a hierarchical/distributed FMS architecture. *Proceedings of the Second International Conference on Computer Integrated Manufacturing*, 538-545.

Rabelo, L., Yih, Y., Jones, A., & Tsai, J.S. (1993). Intelligent scheduling for flexible manufacturing systems. *Proceedings of the IEEE International Conference on Robotics and Automation,* 810-815.

Rachavachari, M. (1988) Scheduling problems with non-regular penalty functions – a review. *Opsearch. 25,* 144-164.

Radhakrishnan, S., & Ventura, J.A. (2000). Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence dependent setup times. *International Journal of Operational Research, 8,* 2233-2252.

Ray, T., & Liew, K. M. (2002). A swarm metaphor for multiobjective design optimization. *Engineering Optimization, 34*, 141-153.

Reisman, A., Kumar, A., & Motwani, J. (1997). Flowshop scheduling/sequencing research: a statistical review of the literature, 1995-1994. *IEEE Transactions on Engineering Management, 44* (3), 316-329.

Reklaitis, G.V. (1992). Overview of scheduling and planning of batch process operations. *Proceeding NATO advanced study institute on batch processing systems engineering, 31*, 660-705.

Rodammer, F.A., & White, K.P. (1988). A Recent Survey of Production Scheduling. *IEEE Transactions on Systems, Man and Cybernetics, 18* (6), 841-851.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning international representations by error propogation. In: D.E. Rumelhart, & J.L. McClelland, (Eds.). *Parallel Distributed : Explorations in the Microstructure of Cognition.* Cambridge: MIT Press.

Sabuncuoglu, I., & Gurgun, B. (1996). A neural network model for scheduling problems. *European Journal of Operational Research, 93*, 288-299.

Sabuncuoglu, I. (1998). Scheduling with neural networks: a review of the literature and new research directions. *Production Planning and Control, 9* (1), 2-12.

Sabuncuoglu, I., & Touhami, S. (2002). Simulation metamodelling with neural networks: an experimental investigation. *International Journal of Production Research, 40* (11), 2483-2505.

Sahni, S.K. (1976). Algorithms for scheduling independent tasks. *J. Assoc. Comput mach., 23*, 116-127.

Salleh, S., & Zomaya, A.Y. (1999). Scheduling in Parallel Computing Systems-Fuzzy and Annealing Techniques. Massachusetts: Kluwer Academic Publishers.

Salman, A., Ahmad, I., & Al-Madani, S. (2002). Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems, 26*, 363-371.

Saratchandran, P., Dundararajan, N., & Foo, S.K. (1996). *Parallel Implementation of Backpropagation Neural Networks on Transputers*. World Scientific.

Satake, T., Morikawa K., & Nakamura, N. (1994). Neural network approach for minimizing the makespan of the general job-shop. *International Journal of Production Economics, 33,* 67-74.

Schaffer, J.D. (Ed.) (1989). *Proceedings of the Third International Conference on Genetic Algorithms*. California: Morgan Kaufmann Publishers.

Schaffer, J.D., Whitley, D., & Eshelman, L.J. (1992). Combinations of genetic algorithms and neural networks : a survey of the state of the art. *Proceedings of International Workshop on Combinations of Genetic Algorithms and Neural Networks*, 1-37.

Schmitz, G.P.J., & Aldrich, C. (1999). Combinatorial evolution of regression nodes in feedforward neural networks. *Neural Networks, 12*, 175-189.

Schwefel, H.P. (1981). *Numerical Optimization of Computer Models*. Chichester: John Wiley & Sons.

Schwefel, H.P. (1995). *Evolution and Optimum Seeking*. New York : John Wiley & Sons.

Sellers, D.W. (1996). A survey of approaches to the job shop scheduling problems. *Proceedings of the 28th Southeastern Symposium on System Theory*, 396-400.

Sendhoff, B., & Kreutz, M. (1999). A model for the dynamic interaction between evolution and learning. *Neural Processing Letters, 10*, 181-193.

Sengor, N.S., Cakir, Y., Guzelis, C., Pekergin, F., & Morgul, O. (1999). An analysis of maximum clique formulations and saturated linear dynamical network. *ARI*, 51, 268-276.

Sethi, R. (1977). On the complexity of mean flow time scheduling. *Mathematics of Operations Research, 2*, 320-330.

Sexton, R., Allidae, B., Dorsey, R., & Johnson, J. (1998). Global optimization for artificial neural networks: A tabu search application. *European Journal of Operational Research, 106* (2-3), 570-584.

Shah, N. (1998). Single and multisite planning and scheduling: Current status and future challenges. *Foundations of Computer Aided Process Operations Proceedings*, 75-90.

Shiue, Y.R., & Su, C.T. (2002). Attribute selection for neural network based adaptive scheduling systems in flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology, 20,* 532-544.

Shiue, Y.R., & Su, C.T. (2003). An enhanced knowledge representation for decision tree based learning adaptive scheduling. *International Journal of Computer Integrated Manufacturing, 16* (1), 48-60.

Sidney, J.B. (1977). Optimal Single Machine Scheduling with Earliness and Tardiness Penalties. *Operations Research*, *25*(1), 62-69.

Sim, S.K., Yeo, K.T., & Lee, W.H. (1994). An expert neural network system for dynamic job shop scheduling. *International Journal of Production Research, 32* (8), 1759-1773.

Sivrikaya-Serifoglu, F.,& Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research*, *26*, 773-787.

Smith, K. (1999). Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research. *Informs Journal on Computing, 11*, 15-34.

Solimanpur, M., Vrat, P., & Shankar, R. (2004). A neuro-tabu search heuristic for the flow shop scheduling problem. *Computers and Operations Research, 31,* 2151-2164.

Sun, H., & Wang, G. (2003). Parallel machine earliness and tardiness scheduling with proportional weights. *Computers & Operations Research, 30,* 801-808.

Sundararaghavan, P., & Ahmed, M.U. (1984). Minimizing the sum of absolute lateness in single-machine and multimachine scheduling. *Naval Research Logistics Quarterly*, *31*, 25-33.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*, 278-285.

Takefuji, Y. (1992). *Neural Network Parallel Computing*. Boston, MA: Kluwer.

Takefuji, Y., Lee, K-C., & Aiso, H. (1992). An artificial maximum neural network: a winner-take-all neuron model forcing the state of the system in a solution domain. *Biological Cybernetics*, *67* (3), 243-251.

Tang, K.S., Chan, C.Y., Man, K.F., & Kwong, S. (1995). Genetic structure for NN topology and weights optimization. *Proceedings of the 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 250–255.

Tasgetiren, M. F., Sevkli, M., Liang, Y. C., & Gencyilmaz, G. (2004). Particle Swarm Optimization Algorithm for Single Machine Total Weighted Tardiness Problem. *Proceedings of the Congress on Evolutionary Computation, 2*, 1412-1419.

Theys, M.D., Braun, T.D., Siegal, H.J., Maciejewski, A.A., & Kwok, Y.-K. (2001). *Mapping Tasks onto Distributed Heterogeneous Computing Systems Using a Genetic Algorithm Approach.* New York: John Wiley and Sons.

Vaithyanathan, S., & Ignizio, J.P. (1992). A stochastic neural network for resource constrained scheduling. *Computers and Operations Research, 19* (3/4), 241-254.

Van Den Bout, D.E., & Miller, T.K. (1988). A Traveling Salesman Objective Function that Works. *Proceedings of IEEE International Conference on Neural Networks*, *2*, 299-303.

Van Hulle, M.M. (1991). A goal programming network for mixed integer linear programming: A case study for the job shop scheduling problem. *International Journal of Neural Systems, 2* (3), 201-209.

Van, Laarhoven, P.J.M., Aarts, E.H.L., & Lenstra, J.K. (1992). Job shop scheduling by simulated annealing. *Operations Research*, *40*, 113-125.

von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik, 15*, 85-100.

Wang, H., Jacob, V., & Roland, E. (2003). Design of efficient hybrid neural networks for flexible flow shop scheduling. *Expert Systems, 20* (4), 208- 231.

Wang, J. (1991). A Time-Varying Recurrent Neural System for Convex Programming. *Proceedings of IJCNN-91-Seattle International Joint Conference on Neural Networks*, 147-152.

Wang, L. (1997). Discrete-Time Convergence Theory and Updating Rules for Neural Networks with Energy Functions. *IEEE Transactions on Neural Networks, 8* (2), 445-447.

Watta, P.B., & Hassoun, M.H. (1996). A Coupled Gradient Network Approach for Static and Temporal Mixed-Integer Optimization. *IEEE Transactions on Neural Networks*, *7*, 578-593.

Werbos, P.J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences.* Ph.D. thesis, Cambridge, MA: Harvard University.

Wicker, D., Rizki, M.M., & Tamburino, L.A. (2002). E-Net: Evolutionary neural network synthesis. *Neurocomputing, 42*, 171-196.

Willems, T. M., & Brandts, E.M.W. (1995). Implementing heuristics as an optimization criterion in neural networks for job-shop scheduling. *Journal of Intelligent Manufacturing, 6*, 377-387.

Willshaw, D.J., & von der Malsburg, C. (1976). How patterned neural connections can be set up by self-organization*. Proceedings of the Royal Society of London, B194*, 431-445.

Yang, S., & Wang, D. (2000). Constraint satisfaction adaptive neural network and heuristics combined approaches for generalized job shop scheduling. *IEEE Transactions on Neural Networks, 11* (2), 474-486.

Yang, S., & Wang, D. (2001). A new adaptive neural network and heuristics hybrid approach for job-shop scheduling. *Computers and Operations Research, 28,* 955-971.

Yang, Y., Kreipl, S., & Pinedo, M. (1997). Heuristics for minimizing the total weighted tardiness in flexible flow shops. *Technical Report*. New York: Stern School of Business, New York University.

Yao, S., Wei, C.J., & He, Z.Y. (1996). Evolving wavelet neural networks for function approximation. *Electronic Letters, 32* (4), 360–361.

Yao, X. (1991). Optimization by genetic annealing. *Proceedings of Second Australian Conference on Neural Networks*, 94-97.

Yao, X. (1997). Global Optimization by evolutionary algorithms. *Proceedings of the IEEE*, 282-291.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE, 87*, 1423-1445.

Yao, X., & Liu, Y. (1998). Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation, 91*, 83-90.

Yoshida, H., Kawata, K.., Fukuyama, Y., & Nakanishi, Y. (1999). A particle swarm optimization for reactive power and voltage control considering voltage stability. *Proceedings of the International Conference on Intelligent System Application to Power System*, 117-121.

Yu, H., & Liang, W. (2001). Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling. *Computers and Industrial Engineering, 39*, 337-356.

Zenter, M.G., & Pekny, J.F. (1994). Learning to solve process scheduling problems: The role of rigorous knowledge acquisition frameworks. In D.W.T. Rippin, J.C. Hale, & J.F.Davis, (Eds.). *Foundations of Computer Aided Process Operations*. CACHE: Austin, TX, 275-309.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning international representations by error propogation. In D.E. Rumelhart, & J.L. McClelland, (Eds.). *Parallel Distributed : Explorations in the Microstructure of Cognition.* Cambridge: MIT Press.

Zhou, D.N., Cherkassy, V., Baldwin, T.R., & Olson, D.E. (1991). A Neural Network Approach to Job-Shop Scheduling. *IEEE Transactions on Neural Networks*, *2*, 175-179.

Zhu, Z., Heady, R.B. (2000). Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach. *Computers & Industrial Engineering, 38*, 297-305.

Zomaya, A.Y., Lee, R.C., & Olariu, S. (2001). An introduction to genetic based scheduling in parallel processor systems. In A.Y. Zomaya, F. Ercal, & S. Olariu, (Eds.). Solutions to Parallel and Distributed Computing Problems (231-254). New York: John Wiley and Sons.