Venkata Sai Dattu Potapragada

A cellular neural network based analog computing approach for ultra-fast adaptive scheduling

Masterarbeit

zur Erlangung des akademischen Grades
Diplom–Ingenieur

Studium Information Technology

Alpen-Adria-Universität Klagenfurt
Fakultät für Technische Wissenschaften

ALPEN-ADRIA
UNIVERSITÄT
KLAGENFURT

Begutachter: Univ.–Prof. Dr. Ing. Kyandoghere Kyamakya

Institut für Intelligente Systemtechnologien
Transportation Informatics

Klagenfurt, im Januar 2009

Eidesstattliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende wissenschaftliche Arbeit selbstständig angefertigt und die mit ihr unmittelbar verbundenen Tätigkeiten selbst erbracht habe. Ich erkläre weiters, dass ich keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle aus gedruckten, ungedruckten oder dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte sind gemäß den Regeln für wissenschaftliche Arbeiten zitiert und durch Fußnoten bzw. durch andere genaue Quellenangaben gekennzeichnet.

Die während des Arbeitsvorganges gewährte Unterstützung einschließlich signifikanter Betreuungshinweise ist vollständig angegeben.

Die wissenschaftliche Arbeit ist noch keiner anderen Prfungsbehörde vorgelegt worden. Diese Arbeit wurde in gedruckter und elektronischer Form abgegeben. Ich bestätige, dass der Inhalt der digitalen Version vollständig mit dem der gedruckten Version bereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

(Unterschrift) (Ort, Datum)

Abstract

Scheduling is the process of allocating resources to the tasks over time. A scheduling problem can be modeled as an assignment problem under constraints and is therefore an optimization problem. In real scenarios a scheduling problem is dynamic in nature as the information required is often gradually revealed or changes over time. In the case of dynamic problems, planning is performed in real time and the related scheduling must be finished before the time window reaches the deadline.

The major factors that make the scheduling problems dynamic are:

- The dynamic nature of tasks: inclusion of new tasks at run-time or the exclusion of some existing tasks results in a need for a rescheduling.
- Random failures: equipment failures result in unpredicted process flows; this generally requires a rescheduling.
- Sequence-dependent setups: makes each job to wait for its time and sometimes delayed because of the eventual unavailability of resources.

Traditional scheduling techniques face a serious complexity problem due to the NP-hard nature of the related optimization problem. Therefore, a key challenge is the search for algorithms/schemes which should provide good quality solutions but at a computation time fitting the requirements of a real-time computing.

The so-called "analog computing" is considered in this work to design a novel scheme capable of delivering an optimal solution of the scheduling problem at a relatively ultra-short time while compared to traditional counterparts' like approaches involving operations research schemes or heuristics. The novel scheme will involve, in the heart, cellular neural networks (CNN) based processors, which will be the key pillar of the analog computing based ultra-fast solver of the dynamic scheduling problems of relevance in transportation and production. CNN-based analog computing has the very interesting advantage of an easy implementation or emulation on digital platforms (i.e., on personal computers or on hardware platforms such as DSP and FPGA boards).

This thesis does in the essence answer the following key questions:

1. What is the potential of the involvement of the analog computing paradigm for solving scheduling problems?
2. How can "scheduling problems" be reformulated and transformed in a set of differential equations in order to allow an easy solution through a processing concept and system involving CNN-based analog computing processors?
3. How far does the "analog computing" based scheduling method involving CNN-processors outperform the traditional approaches based mainly on operations research and neural networks?
4. How scalable is the analog computing based approach? Does it ensure or enable a real-time (re-)scheduling capability?

To find appropriate answers to the research questions formulated above the different works of this thesis have been articulated around the following overall objectives:

- Learning basics of the analog computing.
- Learning basics of cellular neural networks and of how CNN can be used to solve nonlinear differential equations.
- Refreshing the basics of both mathematical programming (i.e. linear programming, nonlinear programming, integer programming, etc.) and scheduling problems.
- Development of a novel approach, based on analog computing, to solve scheduling problems by CNN-processors.
- Implementation and validation through a series of experiments of the novel CNN-based analog computing approach for ultra-fast solution of scheduling problems.
- Finally, asses the superiority (i.e. how much faster it is) and the scalability of the CNN-based analog computing approach when compared to traditional methods.

The main contribution of this thesis is the development and validation of a consistent methodology for transforming and reformulating any scheduling problem into a set of differential equations. It is well known from recent literature that CNN processors are capable of solving ordinary and partial differential equations very efficiently. Thus, the next logical step is to design and implement appropriate CNN processor architectures to solve the differential equations obtained.

To transform a scheduling problem into a set of differential equations we first formulate it as a mathematical programming problem, for example linear/nonlinear programming. Then we use the "Lagrangian relaxation method" to relax the equations from constraints. Next, the relaxed problem is then converted into differential

equations by applying the so-called “neuron dynamics”. The differential equations obtained are solved using CNN processors. In this work, the CNN processors have been implemented in Simulink; however, a hardware implementation on DSP or FPGA is also thinkable and planned for the future.

Diverse scenarios have been considered in the experimental and validation part of this thesis: (a) simple combinatorial optimization problems; (b) extension to some scheduling problems. The results have been compared with other approaches and published works from literature. The comparison criteria have been the exactitude of the results and the computation speed. It could clearly been demonstrated that the CNN-based analog computing approach for scheduling is ultra-fast and produce excellent results when compared to other traditional approaches (for example, just for illustration, a speed-up of more than 200 for a scheduling involving 10 jobs could be observed). The scalability of the novel approach could also be confirmed, since its computational time increases linearly with the problem size, whereas all traditional ones generally display an exponential increase of the computational time.

Keywords: *Analog computing, Scheduling, Cellular Neural Networks, Mathematical programming.*

Acknowledgments

First and foremost I would like to thank Univ. Prof. Dr. Ing. Kyandoghere Kyamakya, for his immense support throughout my thesis with patience. I am extremely grateful to my supervisor, M.Sc Koteswara Rao Anne for his support and advice throughout my thesis. I would like to thank Dr. Do Trong Tuan and Dr.-Ing. Jean Chamberlain Chedjou for reviewing my thesis and for the valuable advices.

The informal support and encouragement of many friends has been indispensable, and I would like particularly to acknowledge the contribution of Sreeram Kumar Baghavatual, Bala Subramanya Reddy Jaggavarapu, Vamsi Prakash Makkapati, Kumara Sri Krishna Mourya Chintalapati, Abin Thomas Mathew and Ranga Raj Gupta Singanamallu.

Finally, I thank my parents, Venkateswara Rao and Sakuntala, who deserve a special word of gratitude and dedication for their moral, emotional, financial and unconditional support. I also thank my sisters, brothers and in-laws who encouraged me positively and showed belief in me. I thank my late grandparents who always inspired me to work hard and to be positive at all times.

Contents

List of Figures

List of abbreviations

ATCS	Apparent Tardiness Cost with Setup
EDD	Earliest Due Date
FIFO	First In First Out
LD	Lagrangian Dual
LRNN	Lagrangian Relaxation Neural Network
MINSLACK	Minimum Slack time
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
SC-CNN	State Controlled Cellular Neural Network
SPT	Shortest Processing Time
WIP	Work In Process

Chapter 1

Introduction and motivation

The aim of this chapter is to give the problem statement and the motivation behind this work. This also gives the basic idea behind the scheduling and the scheduling problems.

1.1 Motivation

Nowadays customer expectations are becoming high in terms of quality, cost and delivery time of product in any company. In order to improve the quality and cost-effectiveness company need to think in a technical aspect whereas to improve the on-time delivery it has to consider the organizational aspect. The later controls and organizes production not only at company level, but also in the supply chain process. Our main focus in this work is on the company level issues. Scheduling is needed to improve the customer satisfaction[1].

The main idea behind scheduling is to increase customer satisfaction by providing goods at right time while utilizing the resources properly. These scheduling problems are difficult to solve because of their NP-hard nature. To solve the hard problems so many approaches are specified which are time consuming. This limitation of solving hard problems in reasonable time by existing methods motivated to propose a new approach. One such techniques which handles hard problems effectively is cellular neural networks(CNN). CNN is an analog simulation technique with parallel processing . CNN processors have been used for researching non-equilibrium systems, constructing non-linear systems, dynamic systems, studying emergent chaotic dynamics, generating chaotic signals. The major advantages of CNN are

- Effective to analyze ordinary differential and partial differential equations.
- Much faster when compared to other regularly used methods.
- Can be implemented in hardware which needs very less space and energy.

1.2 Problem statement

Scheduling is for optimal utilization of available resources for tasks and at the same time to provide customer satisfaction. The main idea behind scheduling is to consider the demand, customer requirement, resources availability to produce goods at right time. This scheduling process is done in many fields of sciences such as transportation, production etc,. We will see the problems in transportation and production processes in detail.

1.2.1 Transportation logistics

In transportation logistics scheduling is needed to minimize the transportation cost operating with in supply and demand constraints. Transportation problem is well studied from past four decades where many researchers developed methods for solving the transportation scheduling problem. The standard transportation problem was first presented by Hitchcock in 1940's [2], which can be expressed as minimization of transportation cost for carrying commodities from m sources to n destinations while operating within supply and demand constraints. This problem can be simply modeled as a linear programming problem. Now-a-days the transportation problems are more complex and dynamic which needs new techniques to solve in real time. Transportation problems are also considered as scheduling the vehicles or goods. From that perspective transportation problems can be seen as planning or scheduling problems. The planning is done on time horizon and based on that time window transportation planning problems are categorized.

Transportation planning is distinguished into four different levels as strategic, tactical, operational and real time. These levels are based on the time period on which the planning is done. The strategic level is one where planning is done for several years and similarly tactical level planning is done for few years to months. The best illustration for strategic level is the planning of airports and for tactical level the planning of staff or vehicle on seasonal basis. The operational level of planning is for days or several hours and the real time level is for seconds or minutes as in fire truck assignments. These four levels of planning are done based on the information available for planning [3, 4].

The availability of information is very important to plan a schedule. This information is available at different stages for different problems such as static problems, stochastic problems and dynamic problems. Information in the case of static problems is available well in advance and in stochastic problems the availability of information is at the time of scheduling. In the case of the dynamic problems no information is available prior to the situation but the information is gradually revealed over time. Static problems belong to strategic or tactical level planning whereas stochastic problems are done at operational level. Some times it needs re-planning in stochastic problems when some random events occur. In case of the dynamic problems planning must be done in real time. In order to perform the real time planning, data processing must be fast and accurate.

Most of the problems in transportation are dynamic in nature because of the unpredictable situations such as traffic congestion, random truck or vehicle failure etc. The schedule in dynamic problems must be released before the time window reaches the dead line.

1.2.2 Production scheduling

Since early 1980's the production scheduling task is mainly based on Just-in-time scheduling process. The decision makers in a production plant need to minimize the work in flow and improve the utilization of the machines. But, nowadays batch processing has its significance as it reduces the time and cost when compared to individual job processing. Also the customers are demanding the products with their own choice of features. These customer requirements make the scheduling more complex. There are different views and perspectives on the production scheduling problem based on the complexity [5]. Those views are mainly in the perspective of problem solving, decision making and organizational aspects.

The job which need to be scheduled has its own specifications such as availability for processing, due date, precedence constraints and other technical data. The decision makers have to consider all these aspects to schedule the job. There are much more details to be considered while scheduling along with the given constraints, some of which are given below:

- Dynamic nature of the job: job may need to be accelerated, need to operate on same machine again (re-entrant flow), new jobs to join etc,. This requires rescheduling of the problem, which demands more computational time to produce a feasible solution [6, 7, 1, 8].

- Batch processing: jobs that require same operation can be grouped to process them as a batch [6, 8].

- Preventive maintenance of machines: time should be provided for the maintenance of machines. Some times unexpected machine failures will force a change in the scheduling [9, 1, 10].

- Setup time and auxiliary resources: need of some auxiliary resources and setup process of a machine adds some waiting time to the job which is to be considered for the scheduling of jobs. This time may vary depending on the other jobs [7, 11, 1].

All these factors are to be considered while scheduling a manufacturing plant. These dynamic tasks will force dynamic changes in the scheduling, thus resulting in rescheduling of the tasks in real-time. This will increase the importance of having speed and effective scheduling process. Solving scheduling problems, with all these complexities, is almost not possible in real time. They come under NP-hard class problems which are not solvable in polynomial time. So many approximation algorithms are available to solve these problems up to some extent.

Consider a flow shop with m-machines and n-jobs, here the main objective is to minimize the makespan of the work. This can be done by finding the possible solutions for this problem and finding the optimal solution from the set of those possible solutions. The number of possible solution for a $n \times m$ (n-jobs, m-machines) flow shop problem is give by $(n!)^m$ i.e, for a problem of 20×10 the number of possible solution are 7.2651×10^{183} [6]. Thus solving this problem is not possible in real time. Most of the real world problems are NP-hard which means that we cannot expect a optimum solution in polynomial time [6].

History proves that there are so many analog computing methods which are faster and can obtain optimum solution to NP-hard problems. One of such analog computing methods is CNN. The most interesting fact about CNN is the speed at which the results are obtained. This is evident form the fact that an image processing primitive (edge, texture, etc.) can be calculated for a 64×64 pixel sized image in less than a microsecond with single chip (64×64 analog CNN array) [12]. The main aim of this thesis is to use this potential of CNN for fast computation and solve the scheduling problem. The scheduling problem is modeled as mathematical programming problem, which makes it a possible to solve it using CNN processor.

1.3 State of the art

Scheduling problem are studied and well developed since past 40 years. The major technologies that contributed to solve this problems vary from crude rules to most sophisticated searching algorithms and artificial intelligence. We will see the evaluation of those technologies in solving the problems:

Dispatching heuristics/ priority rules

Dispatching heuristics/ priority rules are most consistently applied to scheduling problems. The main idea is to assign ranks/weights to tasks based on their priority. The rank is considered while processing the task i.e., task with high priority is assigned with highest rank/weight so that it is processed first. The main advantages of these priority rules are: easy to understand, easy to apply and requires less computation time. Though these rules are very effective for solving problem but they failed in dynamic situations. Some well know rules are first-in-first-out (FIFO), shortest setup plus processing time (SSPT), largest processing time (LPT), shortest processing time (SPT) etc,.

Mathematical programming techniques

Many researchers recognized that a scheduling problem can be solved by using mathematical programming techniques. These techniques are applied very extensively to solve scheduling problems. Problems are formulated as integer programming, mixed integer programming or dynamic programming problems. These programming models contains an objective function with constraints. These methods have been extensively used but the NP-hard nature of the problem limited their use. It is also evident that the computation time increases exponentially with linear increase in the problem size. To overcome this limitation methods like decomposition of problem and others are proposed. Still there exists lot of difficulties in solving a scheduling problems using these techniques. Examples of these techniques are Lagrangian relaxation-based approaches, Decomposition methods, Queuing network models etc,.

Neighborhood search methods

Neighborhood search methods are very popular. These methods provide good solutions and are more effective if combined with some heuristic methods. These techniques continue to perturbate and evaluate schedules until there is no further improvement. This helps in the improvement of quality of the solution. The process stops when it reaches the final optimal solution [13]. There are so many algorithms that follow this method. They all have their own perturbing, improving and stopping methods. They also have their own method of avoiding local optimization. The major disadvantage of this approach is that these methods cannot assure global optimization as they get trapped at local optimum. Different approaches that come under this method are Tabu search, Simulated annealing, Genetic algorithm etc,.

Artificial intelligence techniques

Artificial intelligence is a combination of human intelligence and techniques. This can be explained as implementing method that appear intelligent. There are many advantages of artificial intelligence such as it uses both qualitative and quantitative knowledge in the process, they are capable of generating more complex heuristics, the selection of heuristic is done on the basics of entire information about the tasks, they can capture complex relations and contains special techniques for powerful manipulation of the information in a new data structure. Apart from these advantages they face many disadvantage as the time consumption, also the maintenance and replacing of these systems is much more difficult. The system specific nature limits their use to new systems and without checking to the optimality they can not generate feasible solution. The well know artificial intelligence systems are Expert-/knowledge-based systems, Artificial neural networks, Fuzzy logic, Petri net based approaches, Agent based systems etc,.

These scheduling techniques however got a common problem in terms of computation time. The computation time is the most important criteria in real time scenario where schedule is needed in very short time. The real time scenario is dynamic in nature that means it always needs rescheduling when a new task enters into the flow. Thus rescheduling must be done in reasonable time. The scheduling techniques so far evolved takes more computation time which is almost a limitation in real time. In real time the schedule should be released before the time window reaches the deadline.

As an attempt to minimize the computation time we introduce an analog computing method for solving scheduling problems. The main idea is to solve a scheduling problem by using Cellular Neural Networks (CNN). CNN is an analog computing paradigm which handles differential equations effectively.

1.4 Aim of the thesis

The thesis introduces a classical method of scheduling based on CNN processing. It gives a brief introduction about CNN processor and the method of solving scheduling problems using the CNN processor. The thesis approach can be addressed with few questions. The answer for these questions will explain the method proposed in the thesis.

- How to solve a scheduling problem using CNN processor?
  - CNN is known for its application of image processing but it also solves differential equations very effectively. So in order to solve a scheduling problem the problem has to be modeled as a differential equation. This will raise another question.
- How to model a scheduling problem in differential equation form?
  - The production scheduling problem can be solved mathematically by using Linear programming method. So to model the scheduling problems as differential equations the logical step is to transform mathematical programming problem into a set of differential equations. A combinatorial optimization problem is converted into differential form by applying Lagrangian relaxation and adding the dynamics of neural networks. This method is implemented for solving scheduling problems and is explained clearly in later chapters.

1.5 Outline of the thesis

The chapters in the report are organized as follows:

- Chapter 2 introduces the production scheduling problem. What is scheduling, methods of solving a scheduling problem, Different levels and environments of production scheduling are briefly explained in this chapter.
- Chapter 3 will give a detailed introduction of Lagrangian relaxation method. The use of Lagrangian relaxation method to scheduling problem. Particularly it introduces the newly developed Lagrangian relaxation neural networks approach and the procedure of transforming linear programming problem into a set of differential equations.
- Chapter 4 starts with the basics of CNN. Its architecture, mathematical background and applications are explained briefly. It will give a clear methodology to solve differential equations using CNN networks.

- Chapter 5 starts with the implementation of the the developed approach to an optimization problem and comparison results with an existing neural networks model. It follows with the solution for a simple production scheduling problem. Results are explained.
- Chapter 6 concludes the thesis and suggests some future work and improvements needed to the proposed method.

Chapter 2

Scheduling

2.1 Introduction

Scheduling is the process of allocating resources to different tasks and at the end it has to satisfy some objectives. Unknowingly we all use scheduling in our daily life. A student schedules his work based on the deadline of assignments. While going for market we all schedule our route, to minimize the distance we travel and to reach home fast. A professor schedules his class work to cover the specified syllabus in given time. A process manager schedules his work to finish it in given time or to minimize the lateness. All these scheduling of works made on the basis of some conditions or constraints.

For a student the objective is to submit the assignment before deadline. At the same time he has to satisfy constraints like quality of the work. For a professor this constraint can be the understandability of lecture. For a process manager machine capacity, precedence etc. the constraints. So scheduling is defined as a decision making process which has a goal subjected to some decision making constraints.

This shows the importance of scheduling in different tasks of daily life. Scheduling is an important task in a manufacturing plant. Lot of research has been done on this production scheduling problem. This scheduling is done manually in the early days, but day by day the complexity is grown which makes it impossible to do manually. This resulted in the introduction of mathematical approaches. Still there are some limitations in scheduling because of the hard nature of the problem.

Production scheduling is the process of allocating the resources and then sequencing of tasks to produce goods [14]. Allocation and sequencing decisions are closely related and it is very difficult to model mathematical interaction between them. The allocation problem is solved first and its results are supplied as inputs to the sequencing problem. High quality scheduling improves the delivery performance and lowers the inventory cost. They have much importance in this time based competition. This can be achieved when the scheduling is done in acceptable computation time, but it is difficult because of the NP-hard nature and large size of the scheduling problem.

Since early 1980's the scheduling task is mainly based on Just-in-time scheduling process. The decision makers in a manufacturing plant need to minimize the work in flow and improve the utilization of the machines. But nowadays batch processing has its significance as it reduces the time and cost when compared to individual jobs. Also the customers are demanding the products with their own choice of features. These requirements from the customers makes the scheduling more complex. There are different views on the production scheduling problem based on the complexity [5]. Those views are mainly in the perspective of problem solving, decision making and organizational perspectives.

The job which need to be scheduled has its own specifications such as availability for processing, due date, precedence constraints and other technical data. The decision makers have to considering all these things to schedule the job. There are more things to be considered while scheduling along with the given constraints, for example availability of resources (raw materials, auxiliary resource, etc,.). So based on these the scheduling task is done at different levels. In the next section we see different levels of scheduling.

2.2 Notation

Defining terms

- n- number of jobs
- m- number of machines
- p_j- processing time of job j
- C_j- completion time of job j
- w_j- weight of job j
- r_j- release date of job j
- b_j- beginning date of job j
- d_j- due date of job j
- L_i- the lateness of job j; give by $C_j - d_j$
- U_j- 1 if job j is scheduled by $d_j$otherwise 0

Machine environment

- 1- One machine
- P- Parallel identical machines
- Q- Parallel machines of different speeds
- R- Parallel unrelated machines
- O- Open shop
- F- Flow shop
- J- Job shop

Characteristics and constraints

- $pmtn$- job preemption allowed
- r_j- jobs have nontrivial release date
- $prec$- jobs are precedence constrained

Optimality criteria

- $\sum C_j$- sum of completion times
- $\sum w_j C_j$- weighted sum of completion times
- $\sum U_j$- Number of on-time jobs
- $\sum w_j U_j$- weighted number of on-time jobs
- $C_{\max}$- makespan
- $L_{\max}$- maximum lateness over all jobs

Three field notation

A scheduling problem is denoted by a three field notation as $\alpha|\beta|\gamma$, where

- α- denotes the machine environment
- β- denotes the characteristics or constraints
- γ- denotes the optimality criterion

One example for such type of notation is $1|r_j, pmtn| \sum C_j$. It denotes the problem of preemptive scheduling jobs with release date and precedence constraints on one machine with objective of minimizing the makespan.

2.3 Production planning and production scheduling

Production planning is finding rough plans for long time period [15]. Whereas production scheduling is the process of finding detailed dispatching for individual machines. Production scheduling is applied for short time periods. This is the main difference between planning and scheduling. Hence one can say that production scheduling as a high-resolution short term planning. The figure 2.1 shows the clear difference between production planning and production scheduling.

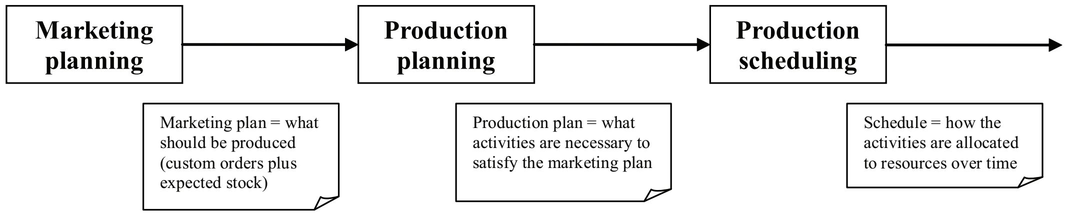

Marketing planning
Production planning
Production scheduling
Marketing plan = what should be produced (custom orders plus expected stock)
Production plan = what activities are necessary to satisfy the marketing plan
Schedule = how the activities are allocated to resources over time

Figure 2.1: Planning and scheduling

A clear understanding of the planning and scheduling is shown as a flow process. In figure 2.2 the process of planning and scheduling is shown. A demand or order is first planned and after that the order is scheduled based on the due dates, material requirements and plant capacity. The scheduled work is then processed on the shop floor.

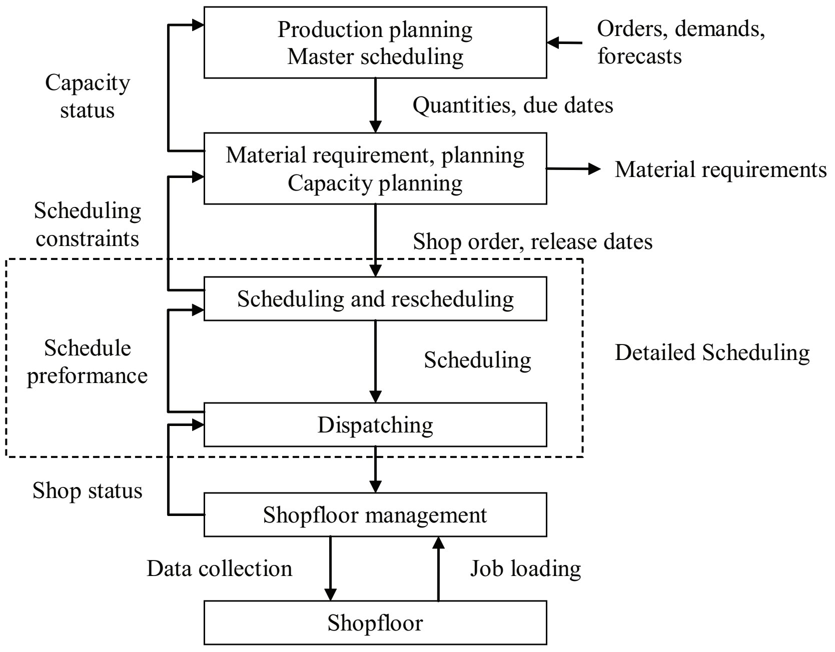


Figure 2.2: Hierarchical planning

In a make-to-order systems order management is a vital issue. Order management is closely related to production capacity, production utilization levels, due date based prizing, and customer priority. Utilization of the resources is based on these factors.

Benefits of production scheduling

The main advantages of production scheduling are:

- Production scheduling determines whether delivery promises can be met and identify time period available for preventive maintenance.
- Production scheduling gives a clear idea to the worker of what is to be done.
- Maximizes the machine/ worker utilization.
- Minimizes the average flow time through the system and setup time.
- It controls the release of jobs to the shop, so ensures raw material availability in time.

These are said to be the main goals of production scheduling. A good scheduling process achieves all these goals.

2.4 Production scheduling environment

There are many types of scheduling environments like small, complex, high-speed, low product variety transfer lines, continuous process flows, etc,. Table2.1 contains a list of different types of environments.

Scheduling environment	Characteristics
Classical job shop	Discrete complex flow, unique jobs, no multie-use parts.
Open job shop	Discrete, complex flow, repetitive jobs and multie-use parts.
Batch shop	Discrete or continuous, less complex flow, many repetitive and multie-use parts, grouping and lotting important.
Flow shop	Discrete or continuous, linear flow, all jobs are highly similar, grouping and lotting important.
Batch/ Flow shop	First half, large continuous batch process; second half a typical flow shop.
Manufacturing cell	Discrete, automated grouped open job shop or batch shop
Assembly shop	Assembly version of open job shop or batch shop
Assembly line	High-volume, low-variety transfer line of assembly shop
Transfer line	Very high-volume, low-variety linear production facility with automated operations.
Flexible transfer line	Modern versions of cell, transfer lines intended to bring some of the advantages of large-scale production to job shop items.

Table 2.1: Scheduling Environment

2.5 Levels of production scheduling

In scheduling, though we model only the decision of what to be released next into the shop and when. There are many more decisions that have to be taken at various stages of manufacturing. A job shop can be considered as a set of resources depending on the current level of abstraction. These decisions are taken step-by-step at an appropriate time and are repeated several times. This process leads to the classification of manufacturing problems at several levels as shown in table 2.2.

Problem Class	Example of Problem	Planning Horizon
Long-range planning	Plant expansion, plant layout, plant design	2-5 years
Middle-.range planning	Production smoothing, logistics	1-2 years
Short-range planning	Requirements plant, shop bidding, due date setting	3-6 months
Scheduling	Job shop routing assembles line balancing, process batch size	2-6 weeks
Reactive scheduling/ control	Hot jobs, down machine, late material	1-3 days

Table 2.2: Levels of scheduling

2.6 Types of production scheduling problem

Based on the machine environment, sequence of operations for the jobs, etc., the production scheduling problem is divided into the different types [16]. These types are explained in the following sections.

One stage, one process or single machine

This is a very simple manufacturing shop type where one machine or process and n-jobs involve. Each job has to go through that single process. This becomes a basic heuristics for complex problem, as a big problem can be divided into small sub problems. This can be solved in real time. There are so many approaches solving this simple scheduling problem.

One stage, multiple processor or parallel machine

This manufacturing shop consists of identical machines may be with different speeds to process jobs. Similar to single machine each job will have only one operation.

Flow shop

This is a typical manufacturing shop. It consists of m-machines and n-jobs. Each job has to go through all the machines in a specific order. It is a unidirectional flow and all the jobs will follow the same steps. The processing time for each job may vary depending up on the job. If there is a job not scheduled on a machine then the process time for the job is considered as zero. Here exists the precedence constraint, that is job $j-1$ must be processed on machine $i-1$ before job j is processed on machine i.

Job shop

Unlike flow shop, job shop is a complex shop where there are finite number of machines, jobs and operation to be done on jobs. There is no direction of flow for jobs. The scheduling is done based on the selection of machine k to process an operation i on job j. Each job can be processed on a machine any number of times.

Static

Here it is assumed that all the jobs are ready for operation at time zero. The resources are available continuously throughout the process without any breakdown. The constraints are same for all the jobs and are well known in advance.

Dynamic

This is the most realistic and complex manufacturing shop when compared to other shops. Here every constraint is unexpected such as release time of jobs is dynamic and random. Resources availability is also dynamic. In order to schedule dynamic shop, we need more effective and short term scheduler which schedules based on the available data for that moment.

2.7 Terminology

Job

A job is the process of obtaining the final product, which consists of many operations.

Machine

A device that executes the required operation for a job. Manufacturing units have different machine environments like single machine, parallel machine and so on.

Arrival time

The time at which a job has to undergo a process.

Processing time

The amount of time required for a job to undergo a task on a machine.

Completion time($C_\mathbf{i}$)

The time at which a job is ready for shipping. That means the time at which processing of a job is finished.

Due date($D_\mathbf{i}$)

Date or time by which the job have be completed.

Lateness

This is to find out how late a job is finished. It is calculated as the difference between due date and completion time. Lateness is mathematically represented as $L_i = C_i - D_i$.

Tardiness

Tardiness of a job T_i is given as $\max(0, C_i - D_i)$.

Preemption

This means splitting of job is allowed during its processing. The unfinished job is done at any time on any machine without loosing the process that have already done.

Setup

It is the process of arranging required auxiliaries or preparing the machine to take-up the job for processing. This is a cause for delay between processing of successive jobs or between the operations of same job. It occurs in many practical units. This time is necessary but unproductive.

2.8 Methods of scheduling

The major aim of the scheduler is to maximize the profit, optimal use of machines available and many more. In order to do scheduling, a manager or a scheduler have to follow techniques. There are many varieties of scheduling techniques. The aim of all the techniques is the same, that is to find a solution for the scheduling problem. Scheduling is done by using many approaches like dispatching rules, mathematical techniques or simulation techniques depending on the complexity of the problem, objectives to reach and so on [17]. Each approach has its own limitations. Obtaining a good solution in relative short time is most important. Based on all these things scheduler follow some techniques. We see about those techniques in the following sections .

- Dispatching heuristics/priority rule
- Mathematical programming techniques
  - Branch and bound method
  - Lagrangian relaxation-based approaches
  - Filtered beam search
  - Decomposition methods
- Neighborhood search methods
  - Tabu search
  - Simulated annealing
  - Genetic algorithm
- Artificial Intelligence (AI) techniques
  - Expert-/knowledge-based techniques systems
  - Artificial neural networks

2.8.1 Dispatching heuristics/priority rule

Dispatching rules are most extensively used for scheduling problem. Scheduling rules and sequencing rules are some of the synonyms used for dispatching rules. In a manufacturing plant the choice of job that is to be processed on a machine is based on the priority of the job. Job with highest priority is selected first. The priority rules are based on the information related to the job that is processing time, due date, arrival time, etc,. There are many priority rules, which are prominent, such as First-In-First-Out (FIFO) based on arrival time, Earliest job Due Date (job-EDD) based on due date, Shortest Processing Time (SPT) based on processing time, MINSLACK based on slack etc,. They are also based on the machine time and their setup times. Some of them are apparent tardiness cost with setup (ATCS), work in process (WIP), etc,.

Research on these dispatching heuristics/priority has been done from many decades to develop many new rules. There are so many other rules that are combination of above mentioned priority rules. Some are weight priority index rules, the jobs are rated based on the weight that is calculated based on the tardiness and thus priority level is defined. All these techniques are widely used as they are easy to understand and produce good solutions. The use of these rules is based on answers to the questions, that mean which rule should we use in order to reach certain performance criterion?

Although these rules have the basic advantages like easy to understand, easy to apply, less computing time, etc,. There are also some disadvantages with these techniques, the main disadvantage is optimal solution cannot be expected. This is not a big problem in simple manufacturing plants, but for a dynamic plant it is difficult to get optimal solution even for single objective.

2.8.2 Mathematical programming techniques

Mathematical programming techniques are widely for production scheduling. The problem formulation is done using linear programming, integer programming or mixed integer programming and dynamic programming methods. There are different approaches in this techniques, the use of these approaches has been limited because of the NP-hardness of the problem . Some of the approaches are explained briefly.

Branch and bound method

Branch and Bound method is a most popular solution technique for integer programming problem. The basic idea of branching is to conceptualize the problem as a decision tree. Each decision choice points a node which corresponds to the partial solution. From each node, there grows a number of new branches, one for each possible decision [18]. This branching process continues until leaf nodes, that cannot branch any further, are reached. These leaf nodes are solutions to the scheduling problems. This method has two steps [19]. The first one is to calculate the lower bounds on the optimal solution and the second is the branching process. The first step guides the branching process. The second step divides a problem into sub problems which are smaller, exhaustive and mutually exclusive. These sub problems are further subdivided. Each branch will continue searching until they get a complete solution. Branching efficiency and choosing the appropriate lower bounds are the fundamental issues in this method. The main disadvantage of this method is the lack of strong lower bound in order to cut branches [20]. Also the computational time of this method is very high for solving large scheduling problems.

Lagrangian relaxation-based approach

Similar to the branch and bound method Lagrangian relaxation-based approach is also a solution technique based on integer programming problem. In Lagrangian relaxation method integer valued constraints are omitted and are added to the objective function with some weight. Then the problem is solved for optimal value. The omitted constraint impose some penalty on the solution if it is not satisfied. Similar to the branch and bound method this is also a solution technique based on integer programming problem. We see about this approach in detail in the coming chapters. The major disadvantage of this approach is the computational time. The time it takes to solve the large scheduling problems limited its usage. This technique is used for two or more objective problems also.

Filtered beam search

This is a special case of branch and bound method. In this method the branches that are to be examined are limited in a intelligent way. Thus the processing time is reduced. Most promising nodes are considered to form branches and weak branch formations are eliminated. Unlike branch and bound method this will guarantee an optimal solution.

Decomposition methods

In this method a large problem is divided into sub problems. Then attempts are made to develop solution for this sub problem which is more easy to understand. Depending on the nature of the problem the solution may be exact or approximate. Here all the sub problems are solved for their optimal values and the solution for the main problem is obtained by reassembling the sub problem solutions [21]. The limitation of this method is that you can not divide all problems into sub problems and sometimes subdivided problems are also be more complex and may need further division.

2.8.3 Neighborhood search method

Neighborhood search methods are very popular. These methods provide good solutions and are more effective if combined with some heuristic methods. These techniques continues to perturbate and evaluate schedules until there is no further improvement. This helps in the improvement of quality of the solution. The process stops when it reaches the final optimal solution [13]. There are so many algorithms that follow this method. They all have their own perturbing, improving and stopping methods. They also have their own method of avoiding local optimization. We see some of the algorithms in the next sections.

Tabu search

The basic idea of tabu search is to explore the search of all feasible scheduling solutions by some sequence of moves [22, 23]. In tabu search method the the moves from one schedule to another schedule is made by evaluating all candidates and choosing the best available ones. Among the moves some are classified as 'tabu'. These moves may lead to cycling or end-up with local minimum. Finally it makes a list of all these moves and the forbidden steps. Based on this list the searching method will be done for other seedlings.

Simulated annealing

As we all know the physical annealing process, the process of cooling and recrystallization of metals, simulation annealing is a analogy of it [24]. Here the energy equation of thermodynamic system is analogous to the objective function. The ground state is for the global minimal. The current state of the thermodynamics is analogous to the scheduling problem. Its application is similar to that of tabu search method, but the selection of neighbor with best objective function value randomizes the selection of the next initial solution [24]. The better the objective value of a neighboring solution, the better chance it stands to be selected as the next starting solution.

Genetic algorithm

Genetic algorithms [25] are an optimization methodology based on a direct analogy to Darwinian natural selection and mutations in biological reproduction. They encode a parallel search through concept space, with each process attempting coarse-grain hill climbing. In genetic algorithms the neighborhood based on a set of schedules. This makes genetic algorithms more powerful in neighborhood search method. The use of genetic algorithm requires five components.

1. A way of encoding solutions to the problem - fixed length string of symbols.

2. An evaluation function that returns a rating for each solution.

3. A way of initializing the population of solutions.
4. Operators that may be applied to parents when they reproduce to alter their genetic composition such as crossover , mutation, and other domain specific operators.
5. Parameter setting for the algorithm, the operators, and so forth.

2.8.4 Artificial intelligence techniques

Artificial intelligence is a combination of human intelligence and techniques. This can be explained as implementing method that appear intelligent. There are four advantages of artificial intelligence

1. Uses both qualitative and quantitative knowledge in the process.
2. Capable of generating more complex heuristics.
3. The selection of heuristic is done on the basics of entire information about the manufacturing plant.
4. They can capture complex relations and contains special techniques for powerful manipulation of the information in a new data structure.

Apart from the above mentioned advantages the main disadvantage is the time consumption. The maintenance and replacing of these systems is much more difficult. The system specific nature limits their use to new systems and without checking to the optimality they can not generate feasible solution. There are so many methods that comes under Artificial Intelligence. We some two of them in the following section.

Expert/knowledge-based systems

Generally these systems consists of two parts: one is knowledge based system and the other one is inference engine to operate on knowledge base. In this knowledge base, all the things that a human memorizes, rules, procedure, heuristics and other abstractions are captured. Then the inference engine selects a strategy to apply to solve the problem. The inference engine can be a forward chaining (data driven) or backward chaining (goal driven).

Artificial neural networks

This is a distributed system approach. This method was motivated by the limitations of the knowledge based systems. The limited knowledge and the ability of solving the problem of a knowledge based system is not applicable for large scheduling problems. Neural networks are developed as an attempt to mimic the learning and prediction ability of a human being [26]. This can be see as a distributed parallel processing model. These models differ in their network topology, learning process, and node characteristics. For example a neural network model can have different types of learning methods like supervised, unsupervised or temporal reinforcement learning. Based on the requirement one selects learning process and network topology. The basic idea is to have a quick learning of good schedule and implement those rules and patterns for other applications.

2.9 Summary

Production scheduling is a major problem in all manufacturing plants. In this chapter we studied the methods of solving production scheduling problem, benefits of production scheduling, different levels and types of production scheduling. These scheduling methods have their own drawbacks, heuristics provide no guaranty for optimization, mathematical methods takes more time for calculation, neighborhood techniques can get trapped in local optima, which limits their application. These drawbacks motivated this thesis work.

Chapter 3

Mathematical programming

3.1 Introduction

Mathematical programming refers to the study of problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from within an allowed set. A mathematical programming problem consists of an objective function and a set of constraints. The aim is to minimize or maximize the objective function while following the constraints. Based on the type of objective function and constraints mathematical programming problems are sub divided as linear programming and nonlinear programming problems. In a linear programming problem the objective function and the constraints are linear. Whereas in a nonlinear programming any one or both, objective function and constraints, are nonlinear. There exists many other types of mathematical programming problems some of them are listed here:

- Linear programming:
- Quadratic programming
- Integer programming
- Nonlinear programming
- Convex programming
- Stochastic programming
- Dynamic programming and etc.

In this chapter we are mainly concentrating on the first two types i.e. on linear programming and quadratic programming problems.

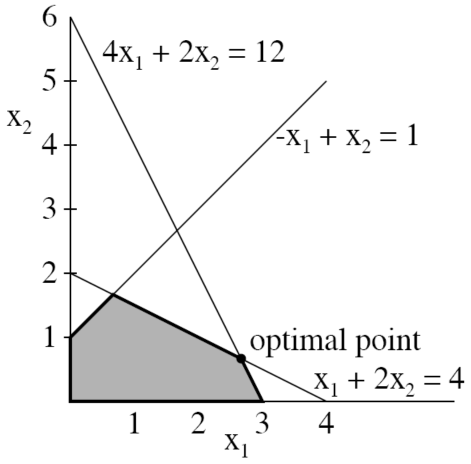


Figure 3.1: Graphical representation of an optimization problem

3.2 Linear programming

Linear programming is a mathematical technique for optimization of a linear objective function, subjected to linear equality and linear inequality constraints. It can be defined as the problem of maximizing or minimizing a linear function subjected to linear constraints. For example

Find numbers x_1 and x_2 that maximizes $x_1 + x_2$ subjected to the constraints $x_1 \geq 0$, $x_2 \geq 0$ and

$$x_1 + 2x_2 \leq 4 \tag{3.1}$$

$$4x_1 + 2x_2 \leq 12 \tag{3.2}$$

$$-x_1 + x_2 \leq 1 \tag{3.3}$$

This problem contains two unknowns and five constraints. All constraints are inequalities and they are all linear in the sense that each involves an inequality in some linear function of the variables. The first two are called 'nonnegativity' constraints and are often found in linear programming. The other three constraints are the 'main constraints'. The function to be minimized is known objective function.Here it is $x_1 + x_2$.

Since there are only two variables, we can solve this problem by graphing the set of points in the plane that satisfies all the constraints (called the constraint set) and then finding which point of this set maximizes the value of the objective function. Each inequality constraint is satisfied by a half-plane of points, and the constraint set is the intersection of all the half-planes. In the present example, the constraint set is the five-sided figure shaded in figure3.1.

We seek the point (x_1, x_2), that achieves the maximum of x_1+x_2 as (x_1, x_2) ranges over this constraint set. The function $x_1 + x_2$ is constant on lines with slope -1, for example the line $x_1+x_2 = 1$, and as we move this line further from the origin up and to the right, the value of x_1+x_2 increases. Therefore, we seek the line of slope -1 that is farthest from the origin and still touches the constraint set. This occurs at the intersection of the lines $x_1+2x_2 = 4$ and $4x_1+2x_2 = 12$, namely,$(x_1, x_2) = (8/3, 2/3)$ The value of the objective function there is $(8/3) + (2/3) = 10/3$.

All linear programming problems are not so easy solve. There may be many variables which are constrained as nonnegative or unconstrained and many main constraints which are equality or inequality constraints. There are two classes of problems one is the standard maximum problem and the other one is the standard minimum problem. In these standard problems all variables are constrained as non negative and all main constraints are inequality constraints.

We are given an m-vector,$b = (b_1, ..., b_m)$, an n-vector, $c = (c_1, ..., c_n)^T$, and an $m \times n$ matrix, of real numbers

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \tag{3.4}$$

- The Standard Maximum Problem: Find an n-vector, $x = (x_1, ..., x_n)^T$, to maximize

$$c^T x = c_1x_1 + ... + c_nx_n \tag{3.5}$$

subjected to the constraints

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \leq b_1 \tag{3.6}$$

$$a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \leq b_2 \tag{3.7}$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \leq b_m \tag{3.8}$$

or

$$Ax \leq b \tag{3.9}$$

and

$$x_1 \geq 0, x_2 \geq 0, ...x_n \geq 0 \tag{3.10}$$

or

$$(x \geq 0) \tag{3.11}$$

- The Standard Minimization Problem: Find an m-vector, $y = (y_1, ..., y_n)^T$, to maximize
$$y^T b = y_1 b_1 + ... + y_n b_n \tag{3.12}$$

subjected to the constraints

$$y_1 a_{11} + y_2 a_{21} + ... + y_m a_{m1} \geq c_1 \tag{3.13}$$

$$y_1 a_{12} + y_2 a_{22} + ... + y_m a_{m2} \geq c_2 \tag{3.14}$$

$$\vdots$$

$$y_1 a_{1n} + y_2 a_{2m} + ... + y_m a_{nm} \geq c_n \tag{3.15}$$

or

$$y^T A \geq b^T \tag{3.16}$$

and

$$y_1 \geq 0, y_2 \geq 0, ...y_n \geq 0 \tag{3.17}$$

or

$$(y \geq 0) \tag{3.18}$$

Note that the main constraints are written as $\leq$ for the standard maximum problem and $\geq$ for the standard minimum problem. The introductory example is a standard maximum problem. We now present examples of four general linear programming problems. These are the most extensively studied problem.

3.2.1 Example 1: The diet problem

The problem is defined as supplying the required nutrients at minimum cost. For this problem we consider that

n - amount of nutrients required for good health,$N_1, ..., N_n$
m - different types of food that supply the nutrients ,$F_1, ..., F_m$,
c_j - the minimum daily requirement of nutrient, N_j
b_i - the cost for unit of food, F_i
a_{ij} - the amount of nutrient N_j contained in one unit of food F_i
y_i- the number of units of food F_i to be purchased per day. To supply the diet

that is essential for a day it costs

$$b_1y_1 + b_2y_2 + ... + b_my_m. \tag{3.19}$$

Then the amount of nutrients contain in the provided diet is give by

$$a_{1j}y_1 + a_{2j}y_2 + ... + a_{mj}y_m. \tag{3.20}$$

for $j = 1, ..., n$. The diet is consider to be perfect unless all the minimum daily requirements are met, That means

$$a_{1j}y_1 + a_{2j}y_2 + ... + a_{mj}y_m \geq c_j \tag{3.21}$$

for $j = 1, ..., n$ We cannot purchase a negative amount of food,

$$y_1 \geq 0, y_2 \geq 0, ...y_m \geq 0 \tag{3.22}$$

Our problem is: minimize 3.19subject to 3.21 and 3.22. This is exactly the standard minimum problem.

3.2.2 Example 2: The transportation problem

The problem is to minimize the transportation cost. where

I - number of ports supply a certain commodity, or production plants, $p_1, ..., P_I$

J - markets to where commodities are shipped,$M_1, ..., M_J$.

Port P_i possesses an amount s_i of the commodity $(i = 1, 2, ..., I)$,

Market M_j must receive the amount r_j of the commodity $(j = 1, ..., J)$.

b_{ij}- the cost of transporting one unit of the commodity from port P_i to market M_j.

y_{ij} - the quantity of the commodity shipped from port P_j to market M_j. Then transportation cost is give by

$$\sum_{i=1}^{I}\sum_{j=1}^{J} y_{ij}b_{ij}. \tag{3.23}$$

The amount of commodity shipped form port P_i is $\sum_{j=1}^{J} y_{ij}$, and the amount of commodity available at port P_i is s_i We can say

$$\sum_{j=1}^{J} y_{ij} \leq s_i \; for \; i = 1, ..., I \tag{3.24}$$

The amount of commodity sent to market M_j is $\sum_{j=1}^{I} y_{ij}$, and the amount required at M_j market is r_j, i.e.,

$$\sum_{i=1}^{I} y_{ij} \leq r_i \; for \; j = 1, ..., I \tag{3.25}$$

We cannot send a negative amount from P_I to M_j, So

$$y_{ij} \geq 0 \; for \; i = 1, ..., I \; and \; j = 1, ..., J \tag{3.26}$$

Our aim is to minimize 3.23, subjected to 3.24,3.25 and 3.26.

3.2.3 Example 3: The activity analysis problem

This problem is to choose the intensities which the various activities are to be operated to maximize the value of the output to the company subject to the given resources.

n - number of activities in a company, $A_1, ...A_n$ using the available supply of
m - number of resources available, $R_1, ...R_m$ (labor hours, steel, etc.).
b_i - the available supply of resource R_i
a_{ij} - the amount of resource R_i used in operating activity A_j at unit intensity.
c_j - the net value to the company of operating activity A_j at unit intensity.
x_j - the intensity at which A_j is to be operated. The value of such an activity allocation is

$$\sum_{j=1}^{n} c_j x_j. \tag{3.27}$$

The amount of resource R_j used in this activity allocation must be no greater than the supply, b_j; that is,

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \; for \; i = 1, ..., m. \tag{3.28}$$

It is assumed that we cannot operate an activity at negative intensity; that is,

$$x_1 \geq 0, x_2 \geq 0, ..., x_n \geq 0. \tag{3.29}$$

Our problem is: maximize3.27 subject to 3.28 and 3.29. This is exactly the standard maximum problem.

3.2.4 Example 4 The optimal assignment problem

This is a problem of choosing a person to a job in order to minimize the total cost/value. For this problem we consider
I - number of persons available for doing jobs
J - number of jobs to be performed

a_{ij}- cost/value of a person I working on job J per day, where $i = 1, ..., I$ and $j = 1, ..., J$ We have to find an assignment, x_{ij}, for$i = 1, ..., I$ and $j = 1, ..., J$ where x_{ij} represents the proportion of time person i to spent on job j. Then,

$$\sum_{j=1}^{J} x_{ij} \leq 1 \quad for\ i = 1, ..., I \tag{3.30}$$

$$\sum_{i=1}^{I} x_{ij} \leq 1 \quad for\ j = 1, ..., J \tag{3.31}$$

and

$$x_{ij} \geq 0 \quad for\ i = 1, ..., I\ j = 1, ..., J \tag{3.32}$$

Our aim is to minimize the total cost/value,

$$\sum_{i=1}^{I} \sum_{j=1}^{J} a_{ij} x_{ij}. \tag{3.33}$$

Constraint 3.30 represents that a person cannot work more than 100% of his time,constraint 3.31 means that only one person can work on a job at a time, and constraint 3.32 says that no one can do a negative amount of work. Finally we have to maximize 3.33 subjected to all the constraints 3.30,3.31 and 3.32

3.2.5 Terminology

- The function to be maximized or minimized is called the objective function.
- The values of x or y is said to be a feasible solution if it satisfies the constraints.
- The set of all feasible values is said to be the constraint set.
- A linear programming problem feasible if and only if the constraint set is not empty; otherwise it is infeasible.

A linear programming problem may be bounded feasible, unbounded feasible or it may be infeasible.

3.3 Quadratic programming

Quadratic programming is a special case of mathematical programming problem with a quadratic objective function to be optimized subjected to a set of linear equality and linear inequality constraints. A quadratic programming problem is formulated as:

$$\min f(x) = CX + \frac{1}{2}X^T Q\, X \tag{3.34}$$

subjected to

$$AX \leq b,\ X \geq 0 \tag{3.35}$$

Where C is an n- dimensional row vector describing the coefficients of the linear terms in the objective function and Q is an $(n \times n)$ symmetric matrix describing the coefficients of the quadratic terms. The constraints are denoted by $(m \times n)$ matrix. A simple numerical example for quadratic programming problem is give as

$$\min 0.5x_1^2 + 0.1x_2^2 \tag{3.36}$$

$$s.t\ x_1 - 0.2x_2 \geq 48 \tag{3.37}$$

$$5x_1 + x_2 \geq 250 \tag{3.38}$$

$$x_1, x_2 \in R \tag{3.39}$$

3.4 Production scheduling problem

Most of the production scheduling problems can be modeled as linear programming problems. The linear program can be solved in polynomial time. To show the process of modeling a problem as a linear program we consider a scheduling problem: The three field notation of the problem is give as

$$R|pmtn|C_{\max} \tag{3.40}$$

This represents the the machine environment is a parallel machines which are not related to each other and the job can be preemptive scheduled. The main objective is to minimize the makespane. In order to explain this modeling we use nmvariables x_{ij} where $1 \leq i \leq m$ and $1 \leq j \leq n$. The variable x_{ij}denotes the fraction of jobs j that is processed on machine i;for example we would interpret a linear programming solution with $x_{1j} = x_{2j} = x_{3j} = \frac{1}{3}$as assigning $\frac{1}{3}$ of job j to machine 1, $\frac{1}{3}$ to machine 2 and $\frac{1}{3}$ to machine 3. Now we study the types of constraints to be assigned to job x_{ij} so that they describe a valid solution to the problem $R|pmtn|C_{\max}$ .It is clear that the fraction of job that is processed on a machine x_{ij} must be a non negative one. We model this constraint as

$$x_{ij} \geq 0 \tag{3.41}$$

We can also say that if a job is completely processed than the sum of fraction of job that is processed on a machine must be 1.

$$\sum_{i=1}^{m} x_{ij} = 1,\ 1 \leq j \leq n \tag{3.42}$$

Our objective is to minimize the makespan $C_{\max}$ of the schedule. Recall that the amount of processing that job j would require if run entirely on machine i is p_{ij}. Therefore for a set of fractional assignments x_{ij} we can determine the amount of time machine i will work: it is just $\sum x_{ij}p_{ij}$, which must be at most $C_{\max}$. We model this with the m constraints

$$\sum_{j=1}^{n} p_{ij}x_{ij} \leq D\ for\ i = 1, ..., m \tag{3.43}$$

Finally we must ensure that no job is processed for more than $C_{\max}$ time. We model this with the n constraints

$$\sum_{i=1}^{m} p_{ij}x_{ij} \leq D,\ 1 \leq i \leq n \tag{3.44}$$

To minimize, we formulate the problem as the following linear program

$$\min\ \ C_{\max} \tag{3.45}$$

$$\sum_{i=1}^{m} x_{ij} = 1,\ for\ j = 1, ..., n \tag{3.46}$$

$$\sum_{j=1}^{n} p_{ij}x_{ij} \leq D\ for\ i = 1, ..., m \tag{3.47}$$

$$\sum_{i=1}^{m} p_{ij}x_{ij} \leq D,\ for\ j = 1, ..., n \tag{3.48}$$

$$x_{ij} \geq 0\ for\ i = 1, ..., m,\ i = 1, ..., m \tag{3.49}$$

It is clear that any feasible schedule for our problem yields an assignment of values to the x_{ij} that satisfies the constraints of our above linear program. But this linear program does not specify the ordering of the jobs on a specific machine but simply assigns the jobs to machines while constraining the maximum load on any machine. In order to specify the order of the jobs we need further modeling of problem. But this will be a Np-hard problem in most of the case. The solution can not be obtained in polynomial time. In the next section we will see some of the relaxations method which resolve a NP-hard problem.

3.4.1 Linear programming relaxation

In order to explain the relaxation methods we consider a linear programming model which is give by $1|r_j, prec|\sum w_jC_j$ .This model is with precedence constrain that means it needs the order of the jobs to be specified. The order of the jobs on a machine is a critical element of high quality solution, so we seek a formulation that can model this problem. We have n variables C_j, one of each job. It represents the completion time of the job in a schedule. The model of this problem is give as

$$\min \sum_j w_jC_j \tag{3.50}$$

subjected to

$$C_j \geq r_j + p_j, j = 1, 2..., n \tag{3.51}$$

$$C_k \geq C_j + p_k \; or \; C_j \geq C_k + p_j \; for \; each \; pair \; j, k \tag{3.52}$$

Unfortunately the last set of constraints are not linear constraints. We use a class of inequality of constraints instead of these introduced by Queyranne [27]. We denote the entire set of jobs as N set of $1, ..., n$ and for any subset $S \subseteq N$ we define $p^2(S) = \sum\limits_j p_j^2$ and $p(S) = \sum\limits_j p_j$. For any feasible one machine schedule we claim that

$$\sum_j p_jC_j \geq \frac{1}{2}(p^2(S) + p(S)^2) \tag{3.53}$$

Now we show that these inequalities are satisfied by the completion times of any valid scheduling for one machine and thus in particular by the completion of a valid schedule for $1|r_j, prec|\sum w_jC_j$ Definition: Let $C_1, ..., C_n$ be the completion times of jobs in any feasible schedule on one machine. Then the $C_j$must satisfy the inequality

$$\sum_j p_jC_j \geq \frac{1}{2}(p^2(S) + p(S)^2) \tag{3.54}$$

Proof: We assume that the jobs are indexed so that $C_1 \leq .... \leq C_n$. Consider the case $S = \{1, ..., n\}$Clearly for any job j, $C_j \geq \sum\limits_{k=1}^{j} p_k$. Multiplying by $p_j$and summing over all j we obtain

$$\sum_{j=1}^{n} p_j C_j \geq \sum_{j=1}^{n} p_j \sum_{k=1}^{j} p_k = \frac{1}{2}(p^2(S) + p(S)^2)$$

It can be applied to any other set of S where we ignore the other jobs. In the case of $1||\sum w_j C_j$ the constraints $\sum_j p_j C_j \geq \frac{1}{2}(p^2(S) + p(S)^2)$ gives an exact characterization of the problem. Specifically any set of C_j that satisfy these constraints must describe the completion times of a feasible schedule. Thus these linear constraints effectively replace $C_k \geq C_j + p_k \ or \ C_j \geq C_k + p_j \ for \ each \ pair \ j, k$. Now we no longer have an exact formulation, but rather a linear programming relaxation of $1|r_j, prec|\sum w_j C_j$.

The main goal of the scheduling problem is to find the beginning time of each job so as to minimize the weighted sum of the completion times. We see the solution method of this problem using CNN in the later chapters.

Chapter 4

Lagrangian relaxation

4.1 Introduction

An optimization problem is a problem of finding an optimal solution from a set of feasible solutions. A combinatorial optimization problem is a typical optimization problem where the feasible solutions can be reduced to discrete set. Generally combinatorial optimization problems are of two categories: one is 'easy' and the other one is 'hard' [28]. The optimization problems that are said to be easy can be solved in real time i.e., polynomial time, while the optimization of hard problems cannot be done in polynomial time. These are known as NP hard problems. There are some pseudo-polynomial algorithms available to solve 'easier hard' optimization problems in polynomial time.

In the early 1970s the optimization problem is represented with an simple objective function subjected to complex constraints. These constraints make the problem hard. In order to solve these hard problems the constraints are relaxed using some relaxation methods. There are so many relaxation methods in use to make a hard problem simple. In the next sections we will see how relaxation is done and what are the properties the relaxed problem must have and how to evaluate the quality of problem.

Notation

If P is an optimization problem, the notation used for it is $FS(P)$, the of feasible solutions of problem P $OS(P)$, the set of feasible solutions of problem P $v(P)$, the set of optimal solution of problem P

4.2 Relaxation method

4.2.1 Relaxation of optimization problem

The relaxation of a minimization problem is formally defined as follows [29]

Definition: Problem $(RP_{\min})$:min$\{g(x)|x \in W\}$ is a relaxation of problem $(P_{\min})$: min$\{f(x)|x \in V\}$, with the same decision variable x, if and only if

1. The feasible set of $(RP_{\min})$contains that of $(P_{\min})$, i.e., $W \supseteq V$, and
2. Over the feasible set of $(P_{\min})$, the objective function of $(RP_{\min})$dominates (is better than) that of $(P_{\min})$, i.e., $\forall x \in V$, $g(x) \leq f(x)$.

If the original problem is a maximization problem then the definition will be
Definition: Problem $(RP_{\max})$:max$\{g(x)|x \in W\}$ is a relaxation of problem $(P_{\max})$: max$\{f(x)|x \in V\}$, with the same decision variable x, if and only if

1. The feasible set of $(RP_{\max})$contains that of $(P_{\max})$, i.e., $W \supseteq V$, and
2. over the feasible set of $(P_{\max})$, the objective function of $(RP_{\max})$dominates (is better than) that of $(P_{\max})$, i.e., $\forall x \in V$, $g(x) \geq f(x)$.

It follows that for the minimization problem $v(RP_{\min}) \leq v(P_{\min})$ and for maximization problem $v(RP_{\max}) \geq v(P_{\max})$, in both cases we can say that $(RP_{\min})$is an optimistic version of $(P_{\min})$. The results in both the cases can be easily translated from one type to the other by remembering that

$$\max \{f(x)|x \in V\} = -\min \{-f(x)|x \in V\} \tag{4.1}$$

The relaxation method provides bounds on the optimal value of difficult problem, and the solution obtained can also be used as a starting point for the specialized heuristics.

4.2.2 Lagrangian relaxation

Lagrangian relaxation is a relaxation technique widely used for the linear programming problems [29]. we will see some characteristics of Lagrangian relaxation method.

Let P is a optimization problem of the form

$$\min \{f(x)|\, Ax \leq b,\, Cx \leq d,\, x \in X\} \tag{4.2}$$

where X may contain non-negative integers or any other set which restricts the solution set of the optimization problem. The constraint $Ax \leq b$ is assumed to be a complicated one. Then the Lagrangian relaxation of P can be given by relaxing the complicated constraint keeping the other constraints unmoved. The Lagrangian

relaxation of P relative to the complicating constraint $Ax \leq b$, with non-negative Lagrangian multiplier λ, is as follows

$$\min\{f(x) + \lambda(Ax \leq b) | \, Cx \leq d, \, x \in X\} \tag{4.3}$$

Here the complicated constraint $Ax \leq b$ is added to the objective function with weight λ and is released from the constraint list. This is known as dualized problem. The problem is relaxation of P since for any x feasible for P and for $\lambda \geq 0$, $f(x) + \lambda(Ax \leq b)$ is less than or equal to $f(x)$.

4.3 Example

We now see the application of Lagrangian relaxation method to an integer problem. This problem is a generalized assignment problem [30, 31].

$$Z = \min \sum_{\mathrm{i}=1}^{\mathrm{m}} \sum_{j=1}^{n} C_{ij} x_{ij} \tag{4.4}$$

$$\sum_{\mathrm{i}=1}^{\mathrm{m}} x_{ij} = 1, \; j = \mathit{1}, ..., n. \tag{4.5}$$

$$\sum_{\mathrm{j}=1}^{\mathrm{m}} a_{ij} x_{ij} \leq b_i, \; i = \mathit{1}, ..., m \tag{4.6}$$

$$x_{ij} = 0 \; \mathit{or} \; 1, \; \mathit{all} \; i \; \mathit{and} \; j \tag{4.7}$$

The natural Lagrangian relaxation for this generalized assignment problem can be obtained in two ways: by relaxing the constraints (4.5) or (4.6).

The dual problem obtained by dualizing the constraint (4.5) is

$$Z_{D1} = \min \sum_{\mathrm{i}=1}^{\mathrm{m}} \sum_{j=1}^{n} C_{ij} x_{ij} + \sum_{j=1}^{n} \lambda_j \left(\sum_{\mathrm{i}=1}^{\mathrm{m}} x_{ij} - 1\right) \tag{4.8}$$

subjected to

$$\sum_{\mathrm{j}=1}^{\mathrm{m}} a_{ij} x_{ij} \leq b_i, \; i = \mathit{1}, ..., m \tag{4.9}$$

$$x_{ij} = 0 \; \mathit{or} \; 1, \; \mathit{all} \; i \; \mathit{and} \; j \tag{4.10}$$

We can rewrite the above dual problem as

$$Z_{D1} = \min \sum_{i=1}^{n} \sum_{\mathrm{j=1}}^{\mathrm{m}} (C_{ij} + \lambda_j) x_{ij} - \sum_{j=1}^{n} \lambda_j \tag{4.11}$$

subjected to

$$\sum_{\mathrm{j=1}}^{\mathrm{m}} a_{ij} x_{ij} \leq b_i,\ i = \mathit{1}, ..., m \tag{4.12}$$

$$x_{ij} = 0 \ \mathit{or}\ 1,\ \mathit{all\ i\ and\ j} \tag{4.13}$$

This problem can be solved in time proportional to $n \sum_{\mathrm{i=1}}^{\mathrm{m}} b_i$ as it resembles the knapsack problem.Another dual problem is obtained by dualizing constraint (4.6) is

$$Z_{D2} = \min \sum_{\mathrm{i=1}}^{\mathrm{m}} \sum_{j=1}^{n} C_{ij} x_{ij} + \sum_{i=1}^{n} \lambda_j (\sum_{\mathrm{j=1}}^{\mathrm{n}} a_{ij} x_{ij} - b_i) \tag{4.14}$$

subjected to

$$\sum_{\mathrm{i=1}}^{\mathrm{m}} x_{ij} = 1,\ j = \mathit{1}, ..., n. \tag{4.15}$$

$$x_{ij} = 0 \ \mathit{or}\ 1,\ \mathit{all\ i\ and\ j} \tag{4.16}$$

$$Z_{D2} = \min \sum_{i=1}^{n} \left(\sum_{\mathrm{j=1}}^{\mathrm{m}} (C_{ij} + \lambda_i a_{ij}) x_{ij} \right) - \sum_{i=1}^{m} \lambda_i b_i \tag{4.17}$$

subjected to

$$\sum_{\mathrm{i=1}}^{\mathrm{m}} x_{ij} = 1,\ j = \mathit{1}, ..., n. \tag{4.18}$$

$$x_{ij} = 0 \ \mathit{or}\ 1,\ \mathit{all\ i\ and\ j} \tag{4.19}$$

Now we can observe that the relaxed assignment problem equation (4.17) is a simple multiple choice problem. For both the dual problems $\lambda \geq 0$, which is necessary condition for $Z_D \leq Z$.

4.4 Lagrangian relaxation neural networks

The Lagrangian relaxation neural networks (LRNN), which deals with the dynamics of neurons, is a combination of recurrent neural network optimization idea with Lagrangian relaxation for constraint handling. This method is better explained with an example. To understand the basic idea behind this approach clearly we consider a separable convex programming problem [31].

Let J is a separable convex programming problem

$$\min_{x} \; J \equiv \sum_{i=1}^{I} J_i(x_i) \tag{4.20}$$

subjected to

$$\sum_{i=1}^{I} g_i(x_i) \leq 0 \quad i = 1, ..., I \tag{4.21}$$

where $x_i \in R^{N_i}$is a $N_i \times 1$ continuous decision variable with $\sum_{i=1}^{I} N_i = N$, $\{J_i(x_i)\}$ and $\{g_i(x_i)\}$ are convex and differentiable functions, and I is number of sub-problems. Since both objective function and constraint function are additive, the problem is separable.First we apply Lagrangian relaxation for this minimization problem. As explained earlier it is just moving the constraint function $g_i(x_i) \leq 0$ into the objective function with a non-negative weight, Lagrangian multiplier λ

$$L(\lambda) \equiv \min_{x} \left[\sum_{i=1}^{I} J_i(x_i) + \lambda^T \sum_{i=1}^{I} g_i(x_i) \right] \tag{4.22}$$

The function $L(\lambda)$ is known as Lagrangian dual (LD) function. As the decision

variables are decoupled through the relaxation method, the equation can be written as sub-problems:

$$L_i(\lambda) \equiv \min_{x_i} \left[\sum_{i=1}^{I} J_i(x_i) + \lambda^T \sum_{i=1}^{I} g_i(x_i) \right], \tag{4.23}$$

and

$$L(\lambda) = \sum_{i=1}^{I} L_i(\lambda) \tag{4.24}$$

The dual problem is then given by

$$\mathbf{LD}: \max_{\lambda \geq 0} L(\lambda) \tag{4.25}$$

The dual function is always a convex function.

The main idea behind Lagrangian relaxation neural networks is to create a network and let the negative dual be its energy function. The negative dual will naturally approaches to its minimum, as the network converges and then $x^\star$, a minimum solution, can be found easily. The negative dual is taken from the relaxed function $L_i(\lambda) \equiv \min_{x_i} \left[\sum_{i=1}^{I} J_i(x_i) + \lambda^T \sum_{i=1}^{I} g_i(x_i) \right]$ or from the sub-problem $L_i(\lambda)$. The relaxed problem which is free of constraints can be solved using recurrent neural network. Finally the main idea behind the Lagrangian relaxation neural network is to merge these two constructs, one for the negative dual and the other for the relaxed problem, and let them feed each other and converge simultaneously. The network elements that updates the multiplier and solve sub-problem are known as Lagrangian neurons and decision neurons respectively. The dynamics of Lagrangian relaxation neural networks is described by the following equations

$$\frac{d\lambda_m(t)}{dt} = \begin{cases} 0 & for\ \lambda_m(t) = 0\ and\ \frac{\partial \tilde{L}(\lambda(t),x(t))}{\partial \lambda_m(t)} < 0 \\ \alpha(t)\frac{\partial \tilde{L}(\lambda(t),x(t))}{\partial \lambda_m(t)} & otherwise \end{cases} \tag{4.26}$$

$$\frac{dx(t)}{dt} = -\beta(t)\frac{\partial \tilde{L}(\lambda(t), x(t))}{\partial x(t)} \tag{4.27}$$

with

$$\tilde{L}(t) = \tilde{L}(\lambda(t), x(t)) = \sum_{i=1}^{I} J_i(x_i(t)) + \lambda(t)^T \sum_{i=1}^{I} g_i(x_i(t)) \tag{4.28}$$

Here $\alpha(t)$ and $\beta(t)$ are positive coefficients and can be time varying. The value of $\alpha(t)$ is calculated using the following equation

$$\alpha(t) = \frac{L^* - \tilde{L}(t)}{\parallel \partial\tilde{L}(t)/\partial\lambda(t) \parallel^2} \tag{4.29}$$

and $\beta(t)$ can be a constant.

This LRNN method is utilized in the thesis work to convert the combinatorial optimization problem into a set of ordinary differential equations. These differential equations are solved using cellular neural networks. The method of solving differential equations using CNN is explained in the next chapter.

4.5 Summary

In this chapter the Lagrangian relaxation method is explained briefly. We have not covered the methods of finding the optimal values for Lagrangian multipliers. We also explained a novel method which is a combination of Lagrangian relaxation and neural networks. This LRNN is the base work for the scheduling approach described in this thesis.

Chapter 5

Cellular neural networks

5.1 Introduction

The concept of Cellular Neural Network(CNN), also called Cellular Non-linear Network was introduced in 1988 by Leon O.Chua and Lin Yang [32, 33, 34]. It is an analog cellular computing paradigm. The original idea from Chua was to use an array of simple, non-linearly coupled dynamic circuits to parallely process large amounts of data in real time. It is a large array of nonlinear dynamic systems. It can be identified as the combination of Cellular Automata [35] and Neural networks [36]. CNN has regularly spaced circuit clones, just like in cellular automata, known as cells that are massively aggregated. The remaining sections in this chapter will give a brief idea about the architecture of CNN processor. Different applications of CNN processor are give and the process of implementation is also explained.

5.2 Basics of CNN processor

5.2.1 Cell

The cell is the basic unit of the CNN. Each cell is made of linear capacitor, a nonlinear voltage-controlled current source and a few resistive linear circuit elements as shown in figure5.1. The cell is a first order nonlinear circuit with u_{ij},x_{ij} and y_{ij} as the input, state and output variables of the cell respectively. The output is a nonlinear function of the state and is given as

$$f(x) = y_{ij} = \frac{1}{2}\left(|x_{ij}+1| - |x_{ij}-1|\right) \tag{5.1}$$

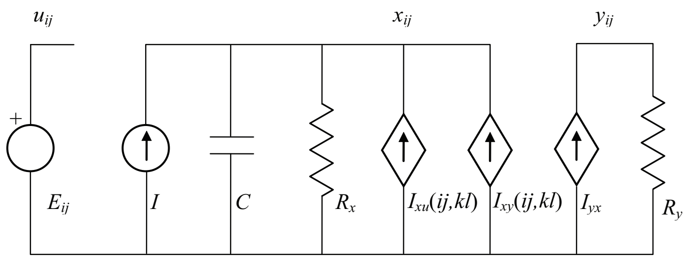


Figure 5.1: CNN cell scheme

All the elements in the basic cell are linear except $I_{yx} = \frac{1}{R_y} f(V_{x_{ij}})$. It is assumed that $|x_{ij}| \leq 1$ as initial constraint condition and the input is a constant and also $|u_{ij}| \leq 1$. Based on the well know Kirchhoff's laws the circuit equation is given as [37]

$$C \frac{dV_x}{dt} = -\frac{1}{R_x} V_x + I + I_{xu} + I_{xy} \tag{5.2}$$

Thus the basic cell is completely defined based on the two equations (5.1) and (5.2)which are the output and the state equations of cell respectively.

5.2.2 CNN architecture

A standard CNN architecture consists of an $M \times N$ rectangular array of cells ($C(i,j)$) with Cartesian coordinates (i,j) where $i = 1, 2, 3, \ldots, M$ and $j = 1, 2, 3, \ldots, N$ as shown in figure5.2. Each cell is connected with its immediate neighbors, i.e., the cells are connected locally, but each cell can also influence more distant cells through propagation. Thus we can say that the cells are globally connected.

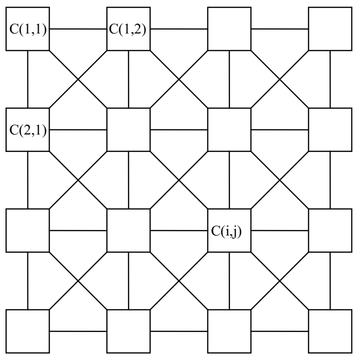


Figure 5.2: CNN architecture

5.2.3 Mathematical foundation

The state equation of a non-isolated cell is give by (5.3), which is clear that the influence of all the other cells exists. The notation used in the equation (5.3) is: i, j represents the Cartesian coordinates of the center cell and k, l represents the others that are influencing the center cell. N_r is the representation for the radius of sphere of influence. figure 5.3 shows the cells which come under the sphere of influence of the center cell. In figure 5.3(a) the sphere of influence is with radius $r = 1$ and figure 5.3(b) with $r = 2$.

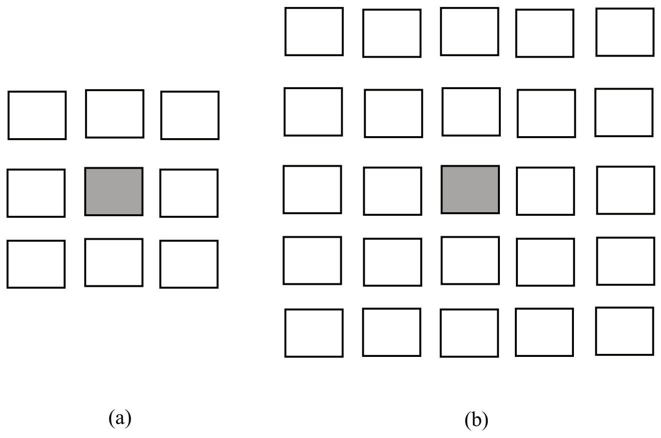


Figure 5.3: Sphere of influence

$$C.\dot{x}_{\mathrm{ij}} = \frac{1}{R} x_{\mathrm{ij}}\,(t) + \sum_{C(k,l)\epsilon N_{\mathrm{r}}(i,j)} A\,(i,j;k,l)\,y_{ij} + \sum_{C(k,l)\epsilon N_{\mathrm{r}}(i,j)} B\,(i,j;k,l)\,u_{\mathrm{kl}} + I \quad (5.3)$$

and the output equation is a piece wise linear equation given as

$$y_{\mathrm{ij}} = \frac{1}{2}\left(|x_{\mathrm{ij}} + 1| - |x_{\mathrm{ij}} - 1|\right) \quad (5.4)$$

The mathematical equation representing the state equation of non isolated cell consists of the $A(i,j;k,l), B(i,j;k,l)$ and I which are

- $A(i,j;k,l)$ - is the Feedback template. The output of the influencing cells($C(k,l) \in N_r(i,j)$) is given to the center cell through this template.
- $B(i,j;k,l)$- is the Control template. The inputs of all the influencing cells ($C(k,l) \in N_r(i,j)$) is give through this template.
- I - is the bias or static template.

The block diagram realization of the mathematical equation is shown in figure 5.4.

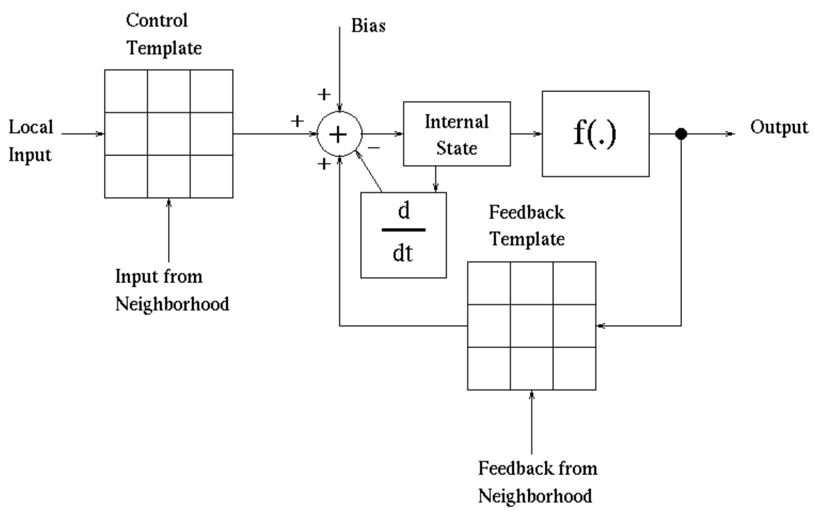


Figure 5.4: Block-scheme of a generical CNN iteration

The feedback and the control templates of a cell with a radius of sphere of influence $r = 1$

$$A = \begin{pmatrix} A(i,j;i-1,j-1) & A(i,j;i-1,j) & A(i,j;i-1,j+1) \\ A(i,j;i,j-1) & A(i,j;i,j) & A(i,j;i,j+1) \\ A(i,j;i+1,j-1) & A(i,j;i+1,j) & A(i,j;i+1,j-1) \end{pmatrix} \tag{5.5}$$

$$B = \begin{pmatrix} B(i,j;i-1,j-1) & B(i,j;i-1,j) & B(i,j;i-1,j+1) \\ B(i,j;i,j-1) & B(i,j;i,j) & B(i,j;i,j+1) \\ B(i,j;i+1,j-1) & B(i,j;i+1,j) & B(i,j;i+1,j-1) \end{pmatrix} \tag{5.6}$$

State-Controlled CNN

A state-controlled cellular neural network (SC-CNN) is a CNN with non-zero linear state feedback template $\hat{A}$. The state equation of a SC-CNN is given below as [37]:

$$\dot{x}_i = -x_i + \sum_{c(j)\in N_r(i)} (\hat{A}_{ij}x_j + A_{ij;}y_i + B_{ij}u_j) + I \tag{5.7}$$

5.2.4 A formal definition

After going through the evolution process of CNN we can now give a formal definition for cellular Neural Networks. Here the last and most general definition of cellular neural network is reported. These definitions were quoted from [37]

Definition 1

(Cellular Neural Networks) A cellular neural network (CNN) is a high dimensional dynamic nonlinear circuit composed by locally coupled, spatially recurrent circuit units called cells. The resulting net may have any architecture, including rectangular, hexagonal, toroidal, spherical and so on. The CNN is defined mathematically by four specifications:

- Cell dynamics.
- Synaptic law.
- Boundary Condition.
- Initial conditions.

Definition 2

(Cell dynamics) The internal circuit core of the cell can be any dynamical system. The cell dynamics is defined by an evolution equation. In the case of continuous-time lumped circuits, the dynamics is defined by the state equation:

$$\dot{x}_\alpha = -g(x_\alpha, z_\alpha, u_\alpha(t), I_\alpha^s) \tag{5.8}$$

where x_α,z_α,$u_\alpha \in R^m$ are the state vector, threshold (bias) and input vector of the cell n_α at position α respectively. I_α^s is a synaptic law and $g:\ R^m \times R^m \times R^m \times R^m \leftarrow R^m$ is a vector field.

Definition 3

(Sphere of influence) The sphere of influence S_α of the cell n_α coincides with the previously defined definition 5.2.4neighbor set N_r without n_α itself:

$$S_\alpha = N_r(n_\alpha) - n_\alpha \tag{5.9}$$

Definition 4

(Synaptic law) The synaptic law defines the coupling between the considered cell n_α and all the cells $n_{\alpha+\beta}$ within a prescribed sphere of influence S_α of n_α itself:

$$I_\alpha^s = \hat{A}_\alpha^\beta x_\alpha + \beta + A_\alpha^\beta f_\beta(x_\alpha, x_{\alpha+\beta}) + B_\alpha^\beta u_{\alpha+\beta}(t) \tag{5.10}$$

The first term is the linear feedback of the states of the neighboring cells. The second term is the nonlinear feedback template. The last term accounts for the contribution of external inputs. B_α^β is the feed-forward or control template.

5.3 Applications of CNN processor

The major application of CNN is in the field of image processing. It also has many other applications. It can be applied for many complex problems. The computational speed of CNN makes it a most competitive tool for the problems where the processing time is a major criteria. Here are some applications of CNN

- Resolution of Complex ODE (ordinary differential equations)
- Resolution of Complex PDE (Models for urban traffic control, heat dissipation, and wave propagation)
- Filtering process in image processing
- Automatic classification of objects
- Biological application
- Medical imaging application
- Implementation of low-pass and high-pass filters
- Morphological operators
- Missile tracking
- Flash detection
- Image segmentation
- Feature extraction
- Color constancy
- Contrast enhancement
- Moving object detection and many more

The interesting application of CNN from this thesis point of view is resolving of complex ODE. In the next section we will see how to implement a CNN for ODE solving.

5.4 Differential equation solving based on CNN processor

A CNN is well known for its image processing application but it is also used to solve non-linear dynamic systems. CNN can be used to solve differential equations of different types. The main step of solving a differential equation is finding the template values by comparing the differential equations with the CNN equations. The basic steps involved in solving the differential equations are

- Basic step, to solve differential equations by using CNN processor, is to transform the set of differential equations into first order differential equations.
- The number of differential, equations that are to be solved, defines the maximum index of the CNN processor array.
- Then the differential equations are arranged in the similar form as CNN processor's state equations. This will make the coefficients comparison of the variables, to the corresponding equation, simple.
- Compare the coefficients of variables to obtain the template values for each CNN processor.
- Use the template values to implement the CNN processor in Matlab/ Simulink.

The most crucial step will be the first one. It is some times very difficult to transform a nonlinear differential equation or a combinatorial optimization problem into a set of first order differential equations.

5.4.1 Example 1

In the previous section 4.4 we have seen the conversion of optimization problem into differential equations. Here we will see an example of solving Chua circuit model using CNN. The reason for choosing Chua's circuit is that; it is the simplest autonomous third-order nonlinear electronic circuit with a rich variety of dynamical behaviors. This helps us in understanding the steps of calculating the template values.

Consider the mathematical model of the Chua's circuit

$$\dot{x} = \alpha\,[y - h(x)] \tag{5.11}$$

$$\dot{y} = x - y + z \tag{5.12}$$

$$\dot{z} = -\beta y + \gamma z \tag{5.13}$$

$$h(x) = m_1 x + \frac{1}{2}(m_0 - m_1)\left(|x+1| - |x-1|\right) \tag{5.14}$$

- The initial step is to transform the given equations into first order differential equations.
  - The Chua circuit model equations are in the same form so we can proceed further.
- Identifying the index of the CNN model. As the number of equations and the variables to be solved are 3. We consider the index of CNN as 3.
- Arranging the given equations similar to CNN equations
  - For our convenience we assume that $x_1 = x$, $x_2 = y$, $x_3 = z$ and $y_i = h(x_i)$. Then the chua circuit equations will be

$$\dot{x}_1 = \alpha\left[x_2 - h(x_1)\right] \tag{5.15}$$

$$\dot{x}_2 = x_1 - x_2 + x_3 \tag{5.16}$$

$$\dot{x}_3 = -\beta x_2 + \gamma x_3 \tag{5.17}$$

$$h(x_1) = m_1 x_1 + \frac{1}{2}(m_0 - m_1)\left(|x_1+1| - |x_1-1|\right) \tag{5.18}$$

- Calculating the template values by comparing the above equations with SC-CNN state equations 5.7.
  - The mathematical model of SC-CNN is given as:

$$\dot{x}_i = -x_i + \sum\left[\hat{A}_{ij}x_j + A_{ij}y_j + B_{ij}u_j\right] + I_i \tag{5.19}$$

$$y_i = h(x_i) = m_1 x_i + \frac{1}{2}(m_0 - m_1)\left(|x_{\mathrm{i}}+1| - |x_{\mathrm{i}}-1|\right) \tag{5.20}$$

i.e.,

$$\dot{x}_1 = -x_1 + \sum_{j=1}^{3}\left[\hat{A}_{1j}x_j + A_{1j}y_j + B_{1j}u_j\right] + I_1 \tag{5.21}$$

$$\dot{x}_2 = -x_2 + \sum_{j=1}^{3}\left[\hat{A}_{2j}x_j + A_{2j}y_j + B_{2j}u_j\right] + I_2 \tag{5.22}$$

$$\dot{x}_3 = -x_3 + \sum_{j=1}^{3} \left[\hat{A}_{3j} x_j + A_{3j} y_j + B_{3j} u_j \right] + I_3 \tag{5.23}$$

$$y_i = h(x_i) = m_1 x_i + \frac{1}{2}(m_0 - m_1)\left(|x_\mathrm{i} + 1| - |x_\mathrm{i} - 1|\right) \tag{5.24}$$

Now comparing the coefficients of the three variables we get the template values as

$\hat{A}_{11} = 1 - \alpha m_1$	$\hat{A}_{21} = 1$	$\hat{A}_{31} = 0$
$\hat{A}_{12} = \alpha$	$\hat{A}_{22} = 0$	$\hat{A}_{32} = -\beta$
$\hat{A}_{13} = 0$	$\hat{A}_{23} = 1$	$\hat{A}_{33} = 1 - \gamma$
$A_{11} = \alpha(m_1 - m_0)$	$A_{21} = 0$	$A_{31} = 0$
$A_{12} = 0$	$A_{22} = 0$	$A_{32} = 0$
$A_{13} = 0$	$A_{23} = 0$	$A_{33} = 0$

- Use the obtained template values to simulate the CNN processors on top of simulink.
  - For the simulation purpose we have chosen the system parameters as $\alpha = 9; \beta = 14.286; \gamma = 0; m_0 = \frac{-1}{7}$and $m_1 = \frac{2}{7}$.
  - By modeling the CNN equations, for the template values and system parameters, on top of simulink as shown in figure 5.5 we obtain the graph as shown in figure 5.6.

5.4.2 Example 2

In the above example we have seen a CNN with linear templates. CNN is also having nonlinear templates which makes it powerful framework for general analog array dynamics. In this section we will see an example where we deal with non-linear templates. For this example we have chosen Lorenz dynamics to simulate on CNN. The following are the Lorenz equations:

$$\frac{dx}{dt} = \sigma(y - x) \tag{5.25}$$

$$\frac{dy}{dt} = x(\rho - z) - y \tag{5.26}$$

$$\frac{dz}{dt} = xy - \beta z \tag{5.27}$$

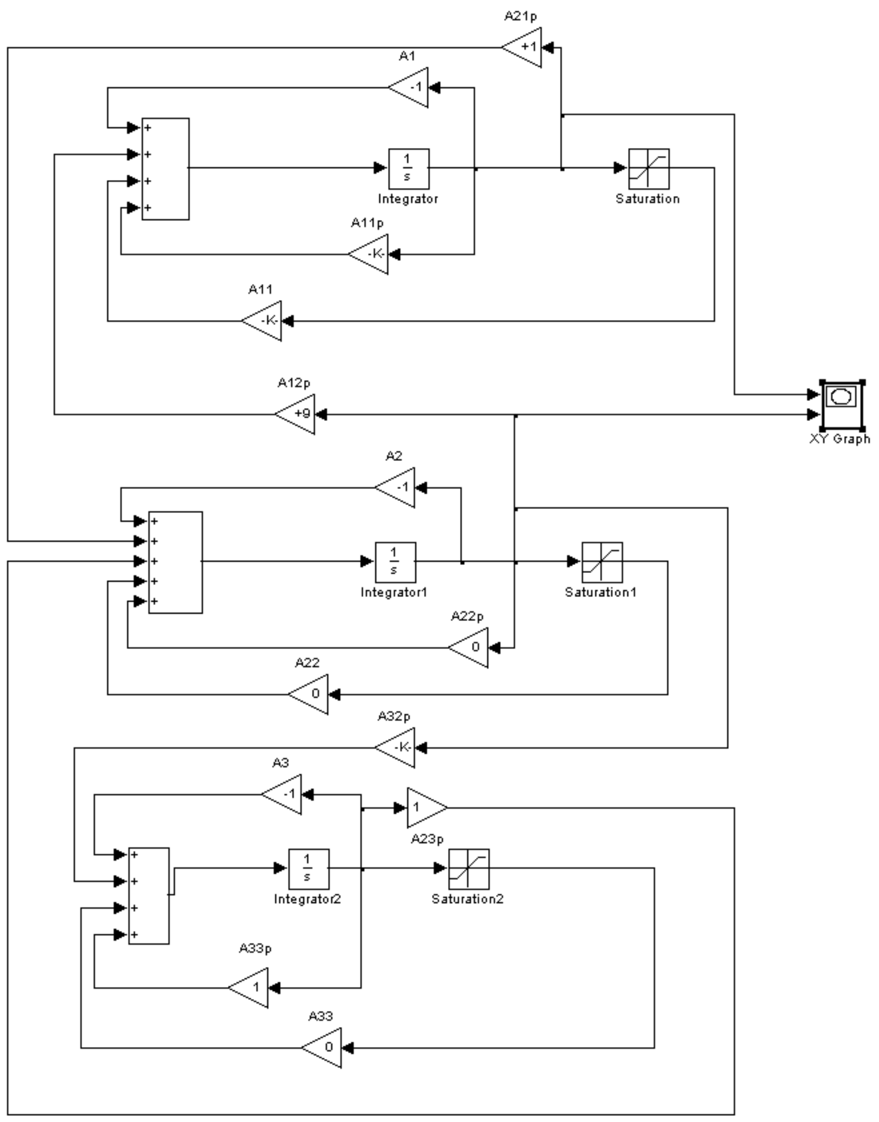


Figure 5.5: Simulink model of Chua's circuit equations

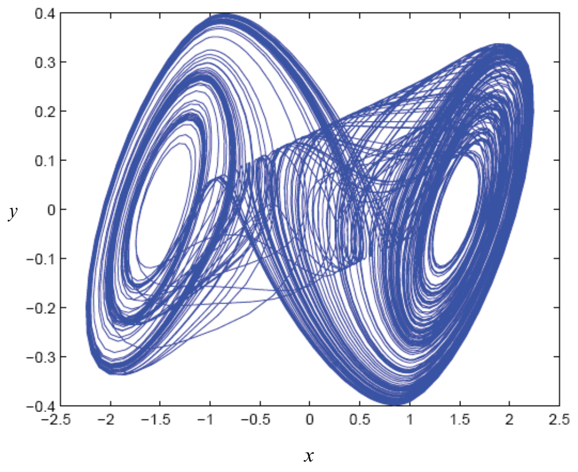


Figure 5.6: Simulation results for Chua's circuit

In order to solve these equations we need to transform them in the form of Chua equations. For this a generalization method is stated in [38]. The type-II generalization equations which are not of first order are written in the form

$$\dot{x} = A(x)(x - F(x)) \tag{5.28}$$

where $A(x)$is an $n \times n$ matrix function of x and $F(x)$ is a mapping of R^n to itself. The matrix A can be chosen as the linearization of the given vector field at the nonzero fixed points. Hence if x_0 is a nonzero fixed point, then matrix is chosen to be the function of this point as $A(x_0)$. similarly F is chosen to be a function of the fixed point, $F(x_0)$.Now by applying the realization to the Lorenz equations we find them in the form:

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{pmatrix} = \begin{pmatrix} -\sigma & \sigma & 0 \\ \rho - z_0 & -1 & x_0 g(x) \\ x_0 g(x) & 0 & -\beta \end{pmatrix} * \begin{pmatrix} x_0 - x_0 g(x) \\ y_0 - y_0 g(y) \\ z_0 - z_0 g(z) \end{pmatrix} \tag{5.29}$$

Now calculate the template values by comparing the equation (5.29) to the SC-CNN equations

$$\dot{x}_i = -x_i + \sum_j \left[\hat{A}_{ij} x_j + A_{ij} y_j + B_{ij} u_j\right] + I_i \tag{5.30}$$

$$y_i = h(x_i) = m_1 x_i + \frac{1}{2}(m_0 - m_1)\left(|x_\mathrm{i} + 1| - |x_\mathrm{i} - 1|\right) \tag{5.31}$$

We get the template values as

$\hat{A}_{11} = -\alpha + 1$ $\quad$ $\hat{A}_{21} = \rho - z_0$ $\quad$ $\hat{A}_{31} = x_0 g(x)$

$\hat{A}_{12} = \alpha$ $\quad$ $\hat{A}_{22} = 0$ $\quad$ $\hat{A}_{32} = 0$

$\hat{A}_{13} = 0$ $\hat{A}_{23} = -x_0 g(x)$ $\hat{A}_{33} = -\beta + 1$

$A_{11} = 0)$ $A_{21} = -\rho x_0 + 2x_0 z_0$ $A_{31} = x_0^2 g(x)$

$A_{12} = 0$ $A_{22} = y_0$ $A_{32} = 0$

$A_{13} = 0$ $A_{23} = 0$ $A_{33} = 0$ and $I_3 = bz_0$ where $y_i = g(x) = \frac{\exp(\gamma x)-1.0}{\exp(\gamma x)+1.0}$ know as sigmoid function

We can observe the non linearity in the template values of $\hat{A}_{23}$,A_{21},$\hat{A}_{31}$.

Now simulating the generalized Lorenz dynamics for the systems parameters $\alpha = -10$, $\beta = 2.67$, $\rho = 28$ and $\gamma = 3$ and for the fixed points $x_0 = y_0 = \pm 8.48$ and $z_0 = 27$ we get the results as shown in figure 5.8. and the simulink model for these nonlinear template values is shown in figure 5.7.

5.4.3 Separable optimization problem solving based on CNN processor

Generally an optimization problem is a set of objective function and some constraints. These problems are solved using linear programming methods. We will now see how to convert a separable optimization problem into first order differential equations.

In the previous chapter we have seen the LRNN approach. In that approach a separable optimization problem is converted into its differential form. In LRNN approach, for the calculation of optimal feasible solution, the dynamics of neurons (Lagrangian neurons and decision neurons) are simulated. This simulation is done by using neural networks. These set of differential equations of Lagrangian neurons and decision neurons are solved using CNN . They can be considered as the dual functions representing the given optimization problem. An example is given to show the method of obtaining the differential equations from a given separable optimization function:

Example

$$\min \frac{1}{2} X^T Q X + C^T X \tag{5.32}$$

subjected to

$$AX = b \tag{5.33}$$

where $X \in R^N$ are continuous decision variables, Q is a positive definite $M \times N$ matrix A is a $M \times N$ matrix Now applying the Lagrangian relaxation method to equations 5.32 and 5.33 the dual function will be

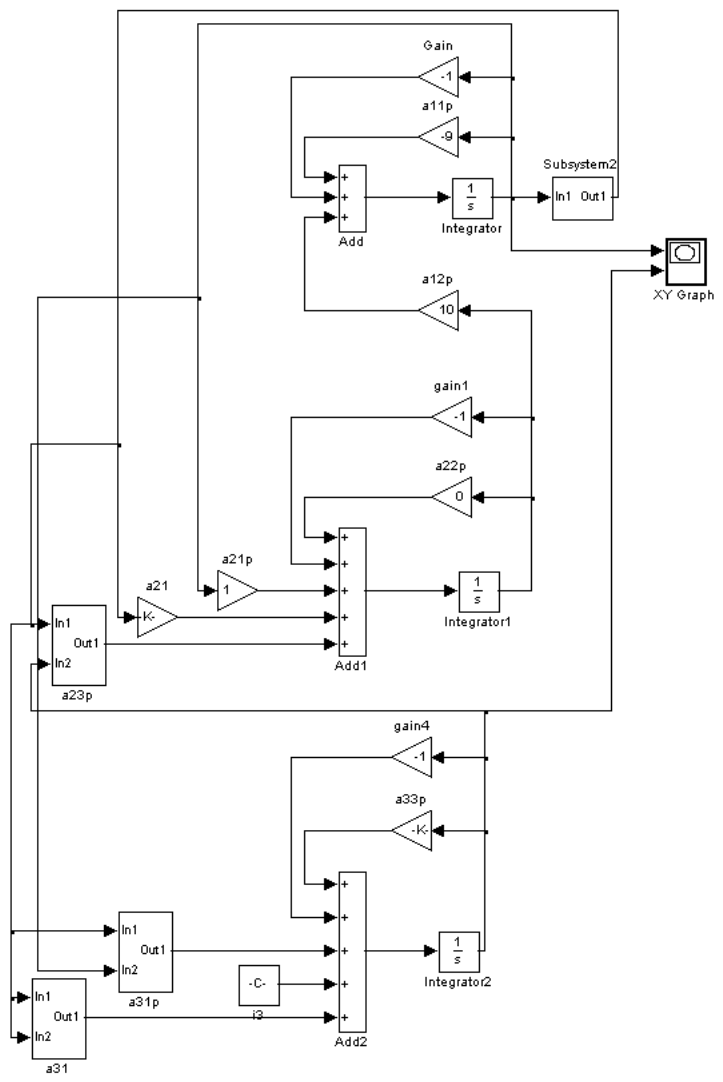


Figure 5.7: Simulink model for Lorenz dynamics

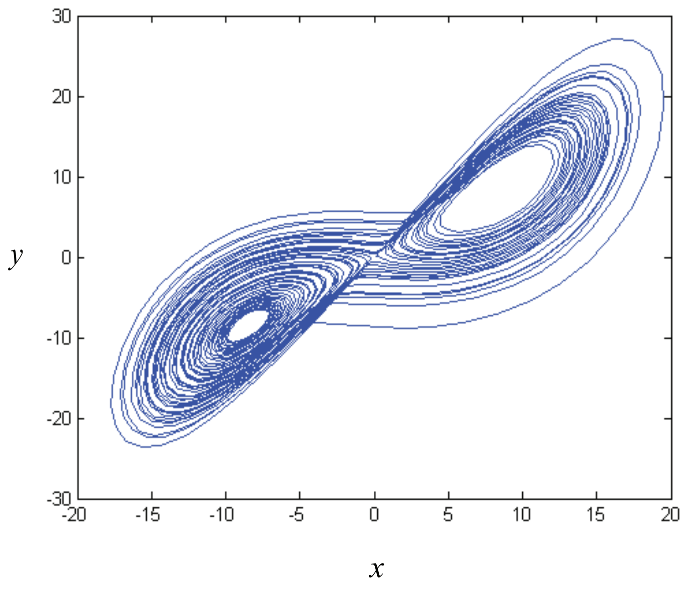


Figure 5.8: Simulation results for Lorenz dynamics

$$L(\lambda, X) = \frac{1}{2}X^TQX + C^TX + \lambda^T(AX - b) \tag{5.34}$$

For Lagrangian neurons and decision neuron of the dual function are:

$$\frac{dX_n}{dt} = -\left[\sum_{j=1}^{N} Q_{nj}X_j + C_n + \sum_{m=1}^{M} A_{mn}\lambda_m, n = 1, 2, ..., N\right], \tag{5.35}$$

and

$$\frac{d\lambda_m}{dt} = \sum_{n=1}^{N} A_{mn}X_n - b_m.m = 1, 2...., M \tag{5.36}$$

Thus a combinatorial optimization problem is transformed into a set of ordinary differential equations. These differential equations of decision neuron and Lagrangian multiplier neuron are solved using CNN. The process of solving the equations is explained in the previous sections. We will see how to resolve the Lagrangian multiplier neurons and decision neuron in the next chapter.

5.5 Summary

In this chapter the basics of CNN are explained and different applications are given. We concentrated mainly on solving differential equations using CNN processor. The method explained in section: 4.4 is used as a base to transform mathematical programming problems into a set of differential equations. Applying these two methods i.e., transforming linear programming problem into a set of differential equations and solving those differential equations using CNN processor, together will be the solution for the scheduling problem. We will see the implementation and results in the later chapters

Chapter 6

Description and implementation

Till now we have seen the method of solving differential equations using CNN processor in section: 5.4. We also studied how to convert an optimization problem into differential form in section: 5.4.3. Now the proposed method is implemented to solve a combinatorial optimization problem and a separable production scheduling problem.

6.1 Description

In order to apply the proposed method the first step is to model a scheduling problem as a mathematical programming problem. This mathematical programming problem is then converted into a set of differential equations. Those differential equations are solved by using CNN processor. The process of transforming mathematical programming problem into a set of differential equations is done in two steps:

1. First step is to transform the scheduling problem, modeled as a mathematical programming problem, into a single equation:

    - This is done by relaxing the constraints i.e., moving the constraints into the objective function by applying the Lagrangian relaxation method.

2. The second step in transforming the mathematical programming problem into a set of differential equations is to find the differential equation for each variable:

    - Partial derivative of the single equation, obtained after applying the Lagrangian relaxation method, is calculated with respect to each unknown in the equation .
    - Then according to the neuron dynamics the polarities are assigned to each variable (differential equation).

- For decision variables the polarity is negative and for Lagrangian multipliers the polarity is positive.

The differential equations obtained from the above step are solved by using CNN processors. The processors of solving the differential equations by using CNN processor is done in few steps:

- The number of differential equations, to be solved, defines the maximum index of the CNN processors array.
- Then the differential equations are arranged in the similar form as CNN processor's state equations. This will make the coefficients comparison of the variables, to the corresponding equation, simple.
- The coefficients of variables are compared, to that of the corresponding CNN processor state equation, to obtain the template values for each CNN processor.
- These template values are used to implement the CNN processor in Matlab/ Simulink.

The simulink model is simulated till a stable optimum result, for each variable involved in the equations, is obtained. The time taken to get a stable optimum result is the computation time taken by CNN processor to generate the result. All the steps explained here are shown in the figure 6.1 for clear understanding.

6.2 Implementation of proposed method

6.2.1 Combinatorial optimization problem

Problem model

Consider the optimization problem shown below

$$\min 0.5x_1^2 + 0.1x_2^2 \tag{6.1}$$

$$s.t\ x_1 - 0.2x_2 \geq 48 \tag{6.2}$$

$$5x_1 + x_2 \geq 250 \tag{6.3}$$

$$x_1, x_2 \in R \tag{6.4}$$

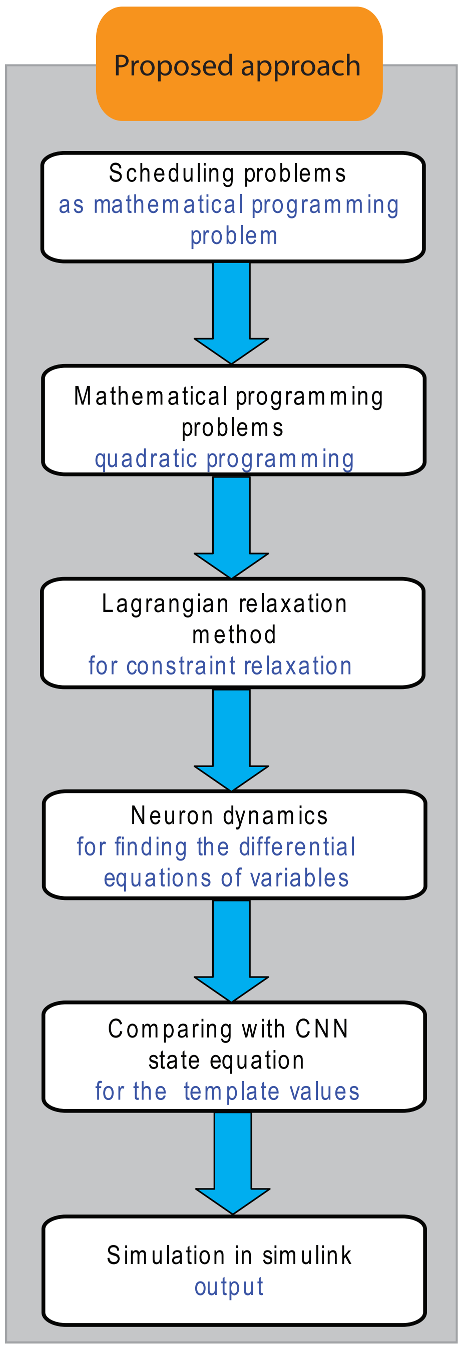

Proposed approach
Scheduling problems
as mathematical programming problem
Mathematical programming problems
quadratic programming
Lagrangian relaxation method
for constraint relaxation
Neuron dynamics
for finding the differential equations of variables
Comparing with CNN state equation
for the template values
Simulation in simulink
output

Figure 6.1: The process to solving scheduling problem using CNN processor

Implementation of proposed method

In order to transform the give quadratic programming problem into a set of differential equations the two constraints are relaxed by using Lagrangian relaxation method

$$\tilde{L}(\lambda_i, x_i) = 0.5x_1^2 + 0.1x_2^2 + \lambda_1(48 - x_1 + 0.2x_2) + \lambda_2(250 - 5x_1 - x_2) \tag{6.5}$$

$$= (0.5x_1^2 - \lambda_1 x_1 - 5\lambda_2 x_1) + (0.1x_2^2 + 0.2\lambda_1 x_2 - \lambda_2 x_2) + 48\lambda_1 + 250\lambda_2 \tag{6.6}$$

The next step is to calculate the Lagrangian neurons and decision neurons. The differential equations, dynamics, for Lagrangian multiplier neurons are given by

$$\frac{d\lambda_1}{dt} = \alpha(t)\frac{\partial \tilde{L}}{\partial \lambda_1} = \alpha(t)(48 - x_1 + 0.2x_2), \tag{6.7}$$

$$\frac{d\lambda_2}{dt} = \alpha(t)\frac{\partial \tilde{L}}{\partial \lambda_2} = \alpha(t)(250 - 5x_1 - x_2). \tag{6.8}$$

and the differential equations for the decision neurons are given as

$$\frac{dx_1}{dt} = -\beta\frac{\partial \tilde{L}}{\partial x_1} = -\beta(x_1 - \lambda_1 - 5\lambda_2), \tag{6.9}$$

$$\frac{dx_2}{dt} = -\beta\frac{\partial \tilde{L}}{\partial x_2} = -\beta(0.2x_2 - 0.2\lambda_1 - \lambda_2). \tag{6.10}$$

The four differential equations are solved for the optimal values. The equations are solved as specified in section:5.4. The four variables λ_1,λ_2,$x_1$and $x_2$is solved by using a 4×4 CNN. The equations are compared with CNN equations and the template values are obtained. The entire process is shown in flowchart for clear understanding. The template values are

$\hat{A}_{11} = 1$	$\hat{A}_{21} = 0$	$\hat{A}_{31} = 1$	$\hat{A}_{41} = 0.2$
$\hat{A}_{12} = 0$	$\hat{A}_{22} = 1$	$\hat{A}_{32} = 5$	$\hat{A}_{42} = 1$
$\hat{A}_{13} = -1$	$\hat{A}_{23} = -5$	$\hat{A}_{33} = 0$	$\hat{A}_{43} = 0$
$\hat{A}_{14} = 0.2$	$\hat{A}_{24} = -1$	$\hat{A}_{34} = 0$	$\hat{A}_{44} = 0.8$

Simulation

A simulink model is formed for the obtained template values as shown in figure 6.3. In this model we consider the nonzero templates only.

Simulinkcombi

The same problem was simulated using a neural networks by Luh et al [31]. The simulation is done using a software and the results were plotted. The results shows the iterations taken by the network to get a stable optimal value. Similar results were obtained using CNN. We will discuss about the results in the next section.

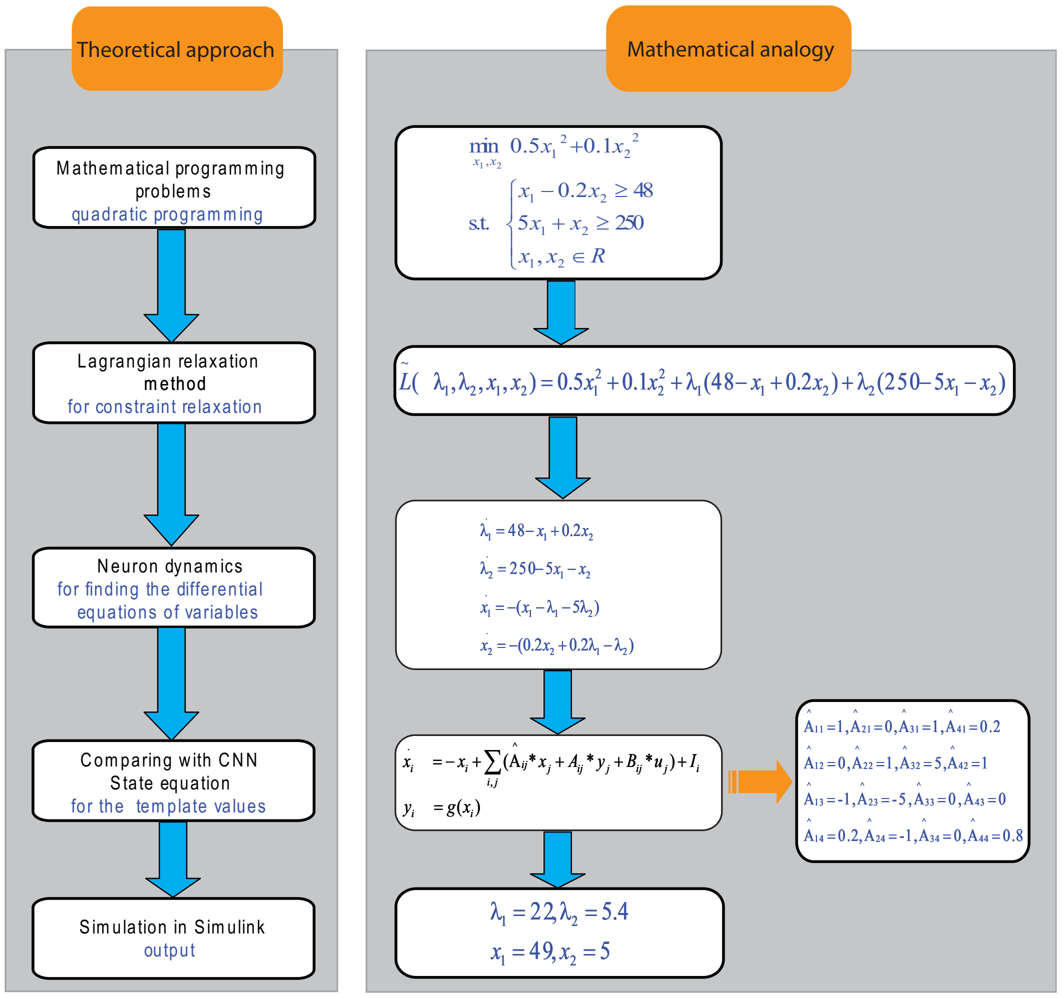

Theoretical approach
Mathematical programming problems
quadratic programming
Lagrangian relaxation method
for constraint relaxation
Neuron dynamics
for finding the differential equations of variables
Comparing with CNN State equation
for the template values
Simulation in Simulink
output
Mathematical analogy
$\min_{x_1,x_2} 0.5x_1^2+0.1x_2^2$
s.t. $x_1-0.2x_2\geq 48$, $5x_1+x_2\geq 250$, $x_1,x_2\in R$
$\tilde{L}(\lambda_1,\lambda_2,x_1,x_2)=0.5x_1^2+0.1x_2^2+\lambda_1(48-x_1+0.2x_2)+\lambda_2(250-5x_1-x_2)$
$\dot{\lambda}_1=48-x_1+0.2x_2$
$\dot{\lambda}_2=250-5x_1-x_2$
$\dot{x}_1=-(x_1-\lambda_1-5\lambda_2)$
$\dot{x}_2=-(0.2x_2+0.2\lambda_1-\lambda_2)$
$\dot{x}_i=-x_i+\sum_{i,j}(\hat{A}_{ij}*x_j+A_{ij}*y_j+B_{ij}*u_j)+I_i$
$y_i=g(x_i)$
$\hat{A}_{11}=1,\hat{A}_{21}=0,\hat{A}_{31}=1,\hat{A}_{41}=0.2$
$\hat{A}_{12}=0,\hat{A}_{22}=1,\hat{A}_{32}=5,\hat{A}_{42}=1$
$\hat{A}_{13}=-1,\hat{A}_{23}=-5,\hat{A}_{33}=0,\hat{A}_{43}=0$
$\hat{A}_{14}=0.2,\hat{A}_{24}=-1,\hat{A}_{34}=0,\hat{A}_{44}=0.8$
$\lambda_1=22,\lambda_2=5.4$
$x_1=49,x_2=5$

Figure 6.2: Implementation of proposed approach to solve an optimization problem

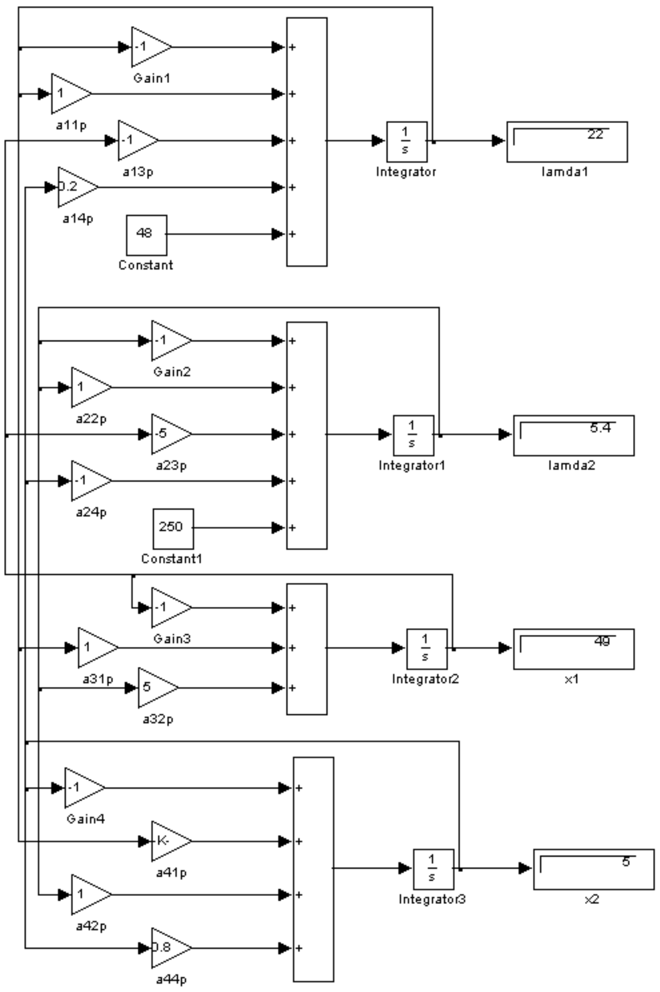


Figure 6.3: Simulink model for the optimization problem

6.2.2 Scheduling problem

In the previous example we have seen the implementation of proposed approach to an optimization problem. In this section we apply the method to a production scheduling problem

Problem model

In order to have clear understanding of the process a simple scheduling problem is selected. We have chosen a $1|r_j|\sum w_jC_j$. The notation represents a one machine environment with the criteria of weighted sum of completions to to be minimized. It is subjected to the release data constraint for each job. The problem model is as shown [39]:

$$\min \sum_j w_jC_j \tag{6.11}$$

$$s.t \sum_j p_jC_j \geq \frac{1}{2}(p^2(S)+p(S)^2) \tag{6.12}$$

$$b_j \geq r_j \tag{6.13}$$

The constraint (6.12) valid inequalities were proposed by Queyranne [27] where $p^2(S)=\sum\limits_j p_j^2$ and $p(S)=\sum\limits_j p_j$. The right hand side of the constraint6.12 is equal to the optimal weighted some of completion times when each job weight is equal to its processing time,min $\sum\limits_j w_jC_j$. The processing time for each job can be considered as weight of the job in normal cases where there are no priority rules to apply. The main variable to be calculated in this scheduling problem is the beginning time of each job b_j, which gives the sequence of jobs. The objective function does not contain this variable directly but we can introduce this variable as $C_j = b_j + p_j$. Thus the given scheduling problem can be written as

$$\min \sum_j p_j(b_j+p_j) \tag{6.14}$$

$$s.t \sum_j p_j(b_j+p_j) \geq \frac{1}{2}(p^2(S)+p(S)^2) \tag{6.15}$$

$$b_j \geq r_j \tag{6.16}$$

Implementation of proposed method

The model of the problem shown in the previous section is a linear programming problem. In order to transform we introduce the Lagrangian multipliers $\lambda 1, \lambda 2_j$ to relax both the constraints 6.15 and 6.16 the dual problem will be:

$$\tilde{L}(\lambda 1, \lambda 2_j, b_j) = \sum_j p_j\,(b_j + p_j) + \lambda 1(\frac{1}{2}(p^2(S) + p(S)^2) - \sum_j p_j(b_j + p_j)) + \lambda 2_j(r_j - b_j) \tag{6.17}$$

where $\tilde{L}(\lambda 1, \lambda 2_j, b_j)$ is know as Lagrangian dual which is free of constraints. Form the dual problem we find the dynamics of Lagrangian multiplier neurons as:

$$\frac{d\lambda 1}{dt} = \alpha(t)\frac{\partial \tilde{L}}{\partial \lambda 1} = \alpha(t)(\frac{1}{2}(p^2(S) + p(S)^2) - \sum_j p_j(b_j + p_j)), \tag{6.18}$$

$$\frac{d\lambda 2_j}{dt} = \alpha(t)\frac{\partial \tilde{L}}{\partial \lambda 2_j} = \alpha(t)(r_j - b_j). \tag{6.19}$$

The dynamic equation 6.19 is a array of i elements where i represents the number of jobs.In order to find the dynamics of decision neurons we need to rewrite the dual equation 6.17 as

$$\tilde{L}(\lambda 1, \lambda 2_j, b_j) = (1 - \lambda 1)\sum_j p_j b_j - \lambda 2_j b_j + \lambda 1(\frac{1}{2}(p^2(S) \\ + p(S)^2) - \sum_j p_j^2) + \lambda 2_j r_j + \sum_j p_j^2 \tag{6.20}$$

In the above equation we can rewrite $\sum_j p_j b_j$ as total sum of product of processing time and beginning times, i.e,

$\sum_j p_j b_j = p_1 b_1 + p_2 b_2 + ... + p_j b_j$. This makes it clear that when we consider first job then we have to deal with the $p_1 b_1$as remaining values are all constants with respect to b_1. So it can be can be stated as for any job j the dynamics of its decision neuron will have the term $p_j b_j$

Now the dynamics of decision neurons are give as

$$\frac{db_i}{dt} = -\beta\frac{\partial \tilde{L}}{\partial b_j} = -\beta(p_j - \lambda 1 p_j - \lambda 2_j), \tag{6.21}$$

These dynamic neuron equations are compared with SC-CNN state equations. The maximum index of the CNN is based on the number of jobs in the manufacturing plant. For general problem, number of jobs j , template values are calculated. The scheduling problem we discussed is based on number of jobs we can calculate the general template values based on the number of jobs. Let n be the number of jobs in the manufacturing plant Then the equations that are to be compared with CNN processor state equation, are (6.18), (6.19) and (6.21) have $2n+1$ variables, one variable to represent $\lambda 1$, n variables to represent $\lambda 2_j$ and similarly n variables to represent

b_j. Thus we need total $2n+1$ SC-CNN state equations. Calculating the template values for these variables will be a laborious process. For the convenience of comparison we assume that variables $x_1, ..., x_n$ represents $b_1, ..., b_n$, $x_{n+1}, ..., x_{n+n}$ represents $\lambda 2_1, ..., \lambda 2_n$ and x_{2n+1}represents $\lambda 1$. Here the templates values for a calculated for general equations with n jobs for the convenience sake we divide the total template values for $1, ..., 2n+1$ into three regions first one is from $1, ..., n$, second region is from $n+1, ..., 2n$ and the third one is $2n+1.\hat{A}_{1,1}, ..., \hat{A}_{2n+1,2n+1}$ i.e., For the region $i = 1, ..., n$, the values are

$$\hat{A}_{i,i} = 1$$

$$\hat{A}_{i,n+i} = \beta(t)$$

$$\hat{A}_{n+i,i} = -\alpha(t)$$

$$\hat{A}_{2n+1,i} = -\alpha(t)p_i$$

$$I_i = -\beta(t)p_i$$

For the region $i = n+1, ..., 2n$

$$\hat{A}_{i,i} = 1$$

$$\hat{A}_{2n+1,i} = 0$$

$$I_i = \alpha(t)r_i$$

For the third region $i = 2n+1$

$$\hat{A}_{2n+1,2n+1} = 1$$

$$I_{2n+1} = \frac{1}{2}\alpha(t)(p(S)^2 - p^2(S))$$

Remaining all template values are zero.

For example the number of jobs are taken as 3 i.e., $n = 3$. The template values for this problem are give as

$$\hat{A}_{11} = 1 \qquad \hat{A}_{22} = 1 \qquad \hat{A}_{33} = 1$$

$$\hat{A}_{14} = 1 \qquad \hat{A}_{25} = 1 \qquad \hat{A}_{36} = 1$$

$$\hat{A}_{17} = p_1 \quad \hat{A}_{27} = p_2 \quad \hat{A}_{37} = p_3$$

$$I_1 = -1 \quad I_2 = -1 \quad I_3 = -1$$

$$\hat{A}_{44} = 1 \quad \hat{A}_{55} = 1 \quad \hat{A}_{66} = 1$$

$$\hat{A}_{41} = -1 \quad \hat{A}_{52} = -1 \quad \hat{A}_{63} = -1$$

$$I_4 = -r_1 \quad I_5 = -r_2 \quad I_6 = -r_3$$

$$\hat{A}_{77} = 1 \quad \hat{A}_{71} = -p_1 \quad \hat{A}_{72} = -p_2$$

$$\hat{A}_{73} = -p_3 \quad I_4 = \tfrac{1}{2}(p(S)^2 - p^2(S))$$

Simulation

The simulation of the scheduling problem with 3 jobs and processing times p1,p2 and p3 is done on top of Simulink. The same model is developed for 6 and 10 jobs. The i in the constant block shows $I_{2n+1} = \frac{1}{2}\alpha(t)(p(S)^2 - p^2(S))$. A simulink model for 3 jobs problem is simulated on op of simulink and the model is as shown in figure 6.4.

6.3 Results

The simulation results obtained for the problems are studied and compared with other existing methods. The results are compared and are explained in this section.

For the combinatorial optimization problem the optimal values for $(x_1, x_2, \lambda_1, \lambda 2)$ are $(49, 5, 22, 5.9)$. The same combinatorial optimization problem is solved by using neural networks, by Luh et al [31]. The trajectory of neurons was studied for each variable and the results were plotted. The graphs shown in the figure 6.5(a) are for the values of variables and the number of iterations taken by neural networks to obtain the optimum result. The graphs show that neural networks take around 2000 iterations to obtain a stable optimum result for the variables.

Similar graphs are plotted for the CNN based solver. The CNN processor takes around 200 iterations to obtain stable optimum results. The graphs plotted for the values and the number of iterations are shown in figure 6.5(b). This comparison shows that the CNN processor takes less iterations when compared to neural networks which results in the reduction of computation time. It demonstrates that the CNN processors are very fast (less computation time) when compared to neural networks.

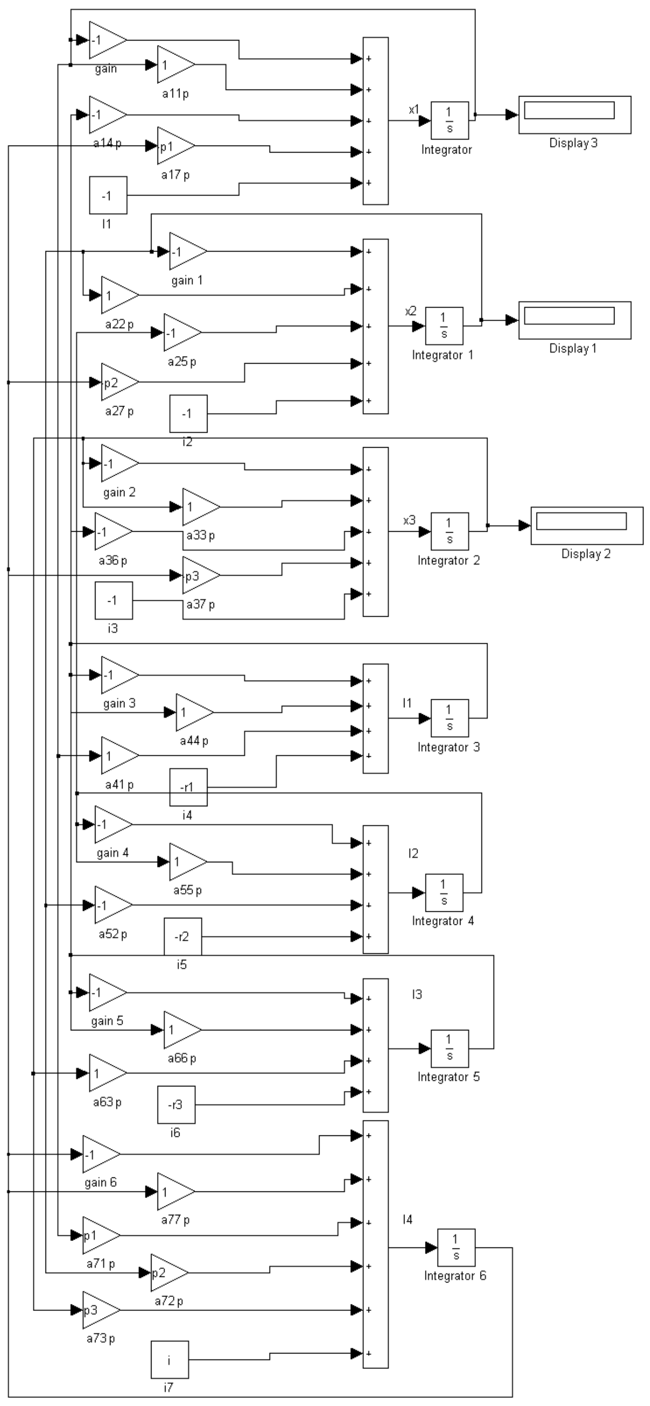
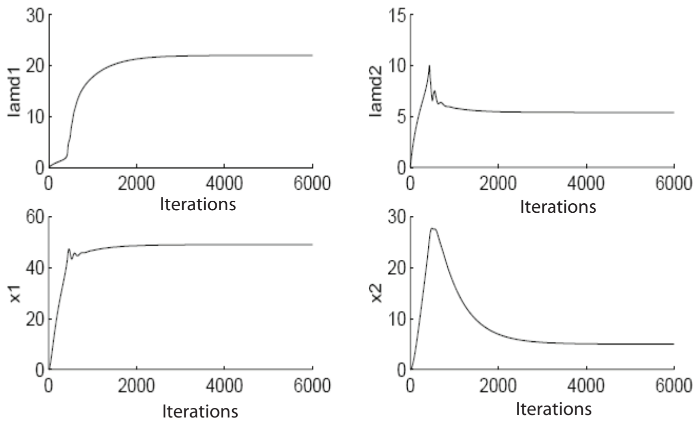
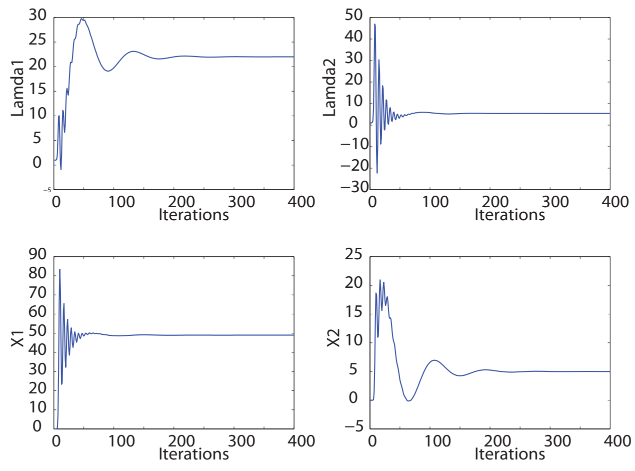


Figure 6.4: Simulation of CNN processors for scheduling problem with 3 jobs on top of Simulink



(a)



(b)

Figure 6.5: (a) Trajectory of neurons for each variable in LRNN, (b) Trajectory of neurons for each variable in CNN processor

Different type of scheduling problems have been solved using the novel CNN based computational approach developed in this thesis, in order to test the proposed method. These results obtained from the CNN based approach are compared with that of the results from traditional operations research based approaches. The proposed approach is tested for scheduling problems with number of tasks varying from 3 to 10, and the computation time to get an optimum plan is compared with that of quadprog and linprog functions (Matlab built-in function for solving optimization problems).

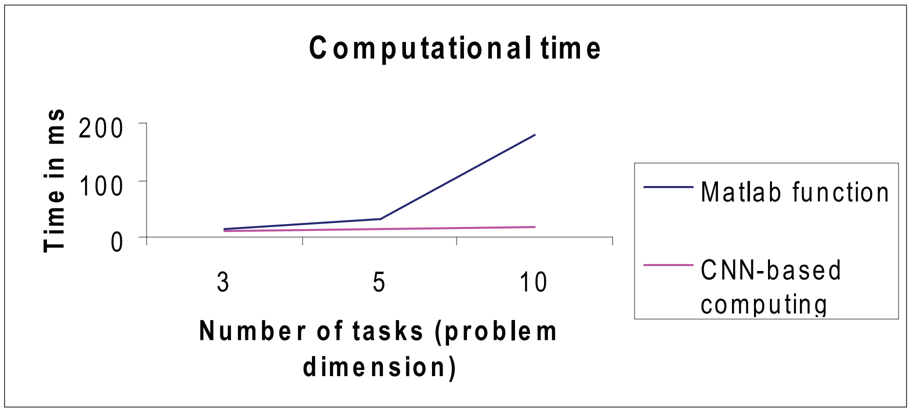


Figure 6.6: Comparison of computation time for CNN based approach and Matlab functions

Computation time required for an optimal scheduling plan for the specific problem with the CNN-based method and Matlab built-in function for solving optimization problems is plotted in the graph shown in figure 6.6. The graph shows that the computation time for CNN-based solver is increasing linearly with the dimension of problem, whereas for the Matlab in-built functions for solving optimization problems, the variation is exponential.

Chapter 7

Conclusion

This thesis in essence has answered the following key questions:

1. How can "scheduling problems" be reformulated and transformed in a set of differential equations in order to allow an easy solution through a processing concept and system involving CNN-based analog computing processors?

2. How far does the "analog computing" based scheduling method involving CNN-processors outperform the traditional approaches based mainly on operations research and neural networks?

3. How scalable is the analog computing based approach? Does it ensure or enable a real-time (re-)scheduling capability?

The main contribution of this thesis is the development and validation of a consistent methodology for transforming and reformulating any scheduling problem into a set of differential equations. It is well known from recent literature that CNN processors are capable of solving ordinary and partial differential equations very efficiently. Thus, the next logical step is to design and implement appropriate CNN processor architectures to solve the differential equations obtained.

To transform a scheduling problem into a set of differential equations we first formulate it as a mathematical programming problem, for example linear/nonlinear programming. Then we use the "Lagrangian relaxation method" to relax the equations from constraints. Next, the relaxed problem is then converted into differential equations by applying the so-called "neuron dynamics". The differential equations obtained are solved using CNN processors. In this work, the CNN processors have been implemented in Simulink; however, a hardware implementation on DSP or FPGA is also thinkable and planned for the future.

Diverse scenarios have been considered in the experimental and validation part of this thesis: (a) simple combinatorial optimization problems; (b) extension to some scheduling problems. The results have been compared with other approaches and published works from literature. The comparison criteria have been the exactitude of the results and the computation speed. It could clearly been demonstrated that the CNN-based analog computing approach for scheduling is ultra-fast and produce excellent results when compared to other traditional approaches (for example, just for illustration, a speed-up of more than 200 for a scheduling involving 10 jobs could be observed). The scalability of the novel approach could also be confirmed, since its computational time increases linearly with the problem size, whereas all traditional ones generally display an exponential increase of the computational time.

Bibliography

[1] C. Artigues, pierre baptiste, and G. bel, *Production scheduling.* John Wiley Sons, Inc. 111 River Street, Hoboken, NJ, 2008.

[2] F. L. Hitchcock, "The distribution of a product from several sources to numerous localities," *J.Math.Phys*, vol. 20, pp. 224–230, 1941.

[3] R. Totschek and R. C. Wood, "An investigation of real-time solution of the transportation problem," *J. ACM*, vol. 8, no. 2, pp. 230–239, 1961.

[4] R. G. kasilingam, *Logistics and Transportation design and planning.* Kluwer academic publishers, 1998.

[5] J. Hermann, "Improving production scheduling," *Integrating Organizational, Decision-Making, and Problem-Solving Perspectives, Industrial Engineering Research Conference*, 2006.

[6] K. Oey and S. J. Mason, "Scheduling batch processing machines in complex job shops," *Proceedings of the 2001 winter simulation conference*, pp. 1200–1207, 2001.

[7] J. W. Herrmann, ed., *Handbook of production scheduling.* Springer Science Business Media, Inc., 2006.

[8] M. Seda, "Mathematical models of flow shop and job shop scheduling problems," *Proceedings of world academy of science, engineering and technology*, vol. 25, pp. 122–127, nov 2007.

[9] X. Yao, "Optimal preventive maintenance scheduling in semiconductor manufacturing," *IEEE transactions on semiconductor manufacturing*, vol. 17, pp. 345–356, Aug 2004.

[10] O. Rose, "Accelerating products under due date oriented dispatching rules in semiconductor manufacturing," *Proceedings of the 2003 winter simulation conference*, pp. 1346–1350, 2003.

[11] I. Habenicht and L. Mnch, "A finite-capacity beam-search-algorithm for production scheduling in semiconductor manufacturing," *Proceedings of the 2002 winter simulation conference*, pp. 1406–1413, 2002.

[12] A. Zarandy, "The art of cnn template design," *International Journal of circuit theory and applications*, vol. 27, pp. 5–23, 1999.

[13] M. Pinedo, "Scheduling, theory, algorithms and systems prentice- hall," *Englewood Cliffs, NJ*, 1995.

[14] G. R. Bitran, D. Maqbool, and L. O. Sison, "A simulation model for job shop scheduling," no. 1402-83., 1983.

[15] R. Bartk, "On the boundary of planning and scheduling," *Study, Proceedings of Eighteenth Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG99) Workshop*, 1999.

[16] Z. P. Bayndr, "Production and distribution systems class notes," *EIN 4333*, 2005.

[17] A. K. Gupta and A. I. Sivakumar, "Job shop scheduling techniques in semiconductor manufacturing," *international Journal Advanced Manufacturing Technology*, vol. 27, pp. 1163–1169, 2006.

[18] T. Morton and D. W. Pentico, *Heuristic scheduling systems.* Wiley-Interscience, 1993.

[19] D. W. Sellers, "A survey of approaches to the job shop scheduling problem," p. 396, 1996.

[20] J. Blazewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: conventionla and new solution techniques," *Europian journal of Operations research*, vol. 93, pp. 1–33.

[21] I. M. Ovacik and R. Uzsoy, *Decomposition methods for complex factory scheduling problems.* Kluwer Academic Publishers.

[22] F. Glover, "Tabu search-part i," *ORSA Journal of Computer*, vol. 1, pp. 190–206, 1989.

[23] F. Glover, "Tabu search-part ii," *ORSA Journal of Computer*, vol. 2, pp. 4–32, 1990.

[24] A. Kiran, "Simulation and scheduling," *Banks J (ed) handbook of simulation, Wiley,Newyork*, pp. 677–717, 1998.

[25] D. Goldberg, "Genetic algorithms in search optimization and machine learning," *Addison-Wesley, Menlo Park, CA*, 1988.

[26] J. Zurada, *Introduction to artificial neural systems.* West Publishing, St. Paul, MN, 1992.

[27] M. Queyranne, "Structure of a simple scheduling polyhedron," *Math. Program.*, vol. 58, no. 2, pp. 263–285, 1993.

[28] M. Fisher, "The lagrangian relaxation method for solving integer programming problems," *Management science*, vol. 27, pp. 1–18, 1981.

[29] M. Guignard, "Lagrangian relaxation," *TOP*, vol. 11, pp. 151–228, Dec 2003.

[30] P. B. Luh, X. Zhao, Y. Wang, and L. S. Thakur, "Lagrangian relaxation neural networks for job shop scheduling," vol. 16, pp. 78–88, Feb. 2000.

[31] P. B. Luh, X. Zhao, Y. Wang, and L. S. Thakur, "Lagrangian relaxation neural networks for job shop scheduling," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1799–1804, 16–20 May 1998.

[32] L. O. Chua and L. Yang, "Cellular neural networks: applications," vol. 35, pp. 1273–1290, Oct. 1988.

[33] L. O. Chua and L. Yang, "Cellular neural networks," *IEEE International Symposium*, vol. 2, pp. 985–988, Jun 1988.

[34] Vandewalle, "Cellular neural networks," *John Wiley & Sons, Inc., New York, NY, USA*, 1994.

[35] S. Wolfram, "Computation theory of cellular automata," *Communications in Mathematical Physics*, vol. 96, pp. 15–57, 1984.

[36] J. Hopeld, "Neural networks and physical systems with emergent collective computational abilities," *PNAS*, vol. 79, pp. 2554–2558, April 1982.

[37] G. Manganaro, P. Arena, and L. Fortuna, *Cellular Neural Networks - Chaos Complexity and VLSI Processing.* Springer, 1998.

[38] R. Brown, "Generalizations of the chua equations," vol. 40, pp. 878–884, Nov. 1993.

[39] F. A. Chudak and D. S. Hochbaum, "A half-integer linear programming relaxation for scheduling precedence-constrained jobs on a single machine," *Oper. Res*, vol. 25, pp. 199–204, 1999.