A Neural Network Model for Resource Leveling

Daniela Savin

A Thesis

in

The Department

of

Centre for Building Studies

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

November 1995

ISBN 0-612-10900-3

Name DANIELA SAVIN

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

APPLIED SCIENCES – ENGINEERING GENERAL | 0537 | U·M·I

SUBJECT TERM | SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture	0729
Art History	0377
Cinema	0900
Dance	0378
Fine Arts	0357
Information Science	0723
Journalism	0391
Library Science	0399
Mass Communications	0708
Music	0413
Speech Communication	0459
Theater	0465

EDUCATION

General	0515
Administration	0514
Adult and Continuing	0516
Agricultural	0517
Art	0273
Bilingual and Multicultural	0282
Business	0688
Community College	0275
Curriculum and Instruction	0727
Early Childhood	0518
Elementary	0524
Finance	0277
Guidance and Counseling	0519
Health	0680
Higher	0745
History of	0520
Home Economics	0278
Industrial	0521
Language and Literature	0279
Mathematics	0280
Music	0522
Philosophy of	0998
Physical	0523
Psychology	0525
Reading	0535
Religious	0527
Sciences	0714
Secondary	0533
Social Sciences	0534
Sociology of	0340
Special	0529
Teacher Training	0530
Technology	0710
Tests and Measurements	0288
Vocational	0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language	
General	0679
Ancient	0289
Linguistics	0290
Modern	0291
Literature	
General	0401
Classical	0294
Comparative	0295
Medieval	0297
Modern	0298
African	0316
American	0591
Asian	0305
Canadian (English)	0352
Canadian (French)	0355
English	0593
Germanic	0311
Latin American	0312
Middle Eastern	0315
Romance	0313
Slavic and East European	0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy	0422
Religion	
General	0318
Biblical Studies	0321
Clergy	0319
History of	0320
Philosophy of	0322
Theology	0469

SOCIAL SCIENCES

American Studies	0323
Anthropology	
Archaeology	0324
Cultural	0326
Physical	0327
Business Administration	
General	0310
Accounting	0272
Banking	0770
Management	0454
Marketing	0338
Canadian Studies	0385
Economics	
General	0501
Agricultural	0503
Commerce-Business	0505
Finance	0508
History	0509
Labor	0510
Theory	0511
Folklore	0358
Geography	0366
Gerontology	0351
History	
General	0578
Ancient	0579
Medieval	0581
Modern	0582
Black	0328
African	0331
Asia, Australia and Oceania	0332
Canadian	0334
European	0335
Latin American	0336
Middle Eastern	0333
United States	0337
History of Science	0585
Law	0398
Political Science	
General	0615
International Law and Relations	0616
Public Administration	0617
Recreation	0814
Social Work	0452
Sociology	
General	0626
Criminology and Penology	0627
Demography	0938
Ethnic and Racial Studies	0631
Individual and Family Studies	0628
Industrial and Labor Relations	0629
Public and Social Welfare	0630
Social Structure and Development	0700
Theory and Methods	0344
Transportation	0709
Urban and Regional Planning	0999
Women's Studies	0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture	
General	0473
Agronomy	0285
Animal Culture and Nutrition	0475
Animal Pathology	0476
Food Science and Technology	0359
Forestry and Wildlife	0478
Plant Culture	0479
Plant Pathology	0480
Plant Physiology	0817
Range Management	0777
Wood Technology	0746
Biology	
General	0306
Anatomy	0287
Biostatistics	0308
Botany	0309
Cell	0379
Ecology	0329
Entomology	0353
Genetics	0369
Limnology	0793
Microbiology	0410
Molecular	0307
Neuroscience	0317
Oceanography	0416
Physiology	0433
Radiation	0821
Veterinary Science	0778
Zoology	0472
Biophysics	
General	0786
Medical	0760

EARTH SCIENCES

Biogeochemistry	0425
Geochemistry	0996
Geodesy	0370
Geology	0372
Geophysics	0373
Hydrology	0388
Mineralogy	0411
Paleobotany	0345
Paleoecology	0426
Paleontology	0418
Paleozoology	0985
Palynology	0427
Physical Geography	0368
Physical Oceanography	0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences	0768
Health Sciences	
General	0566
Audiology	0300
Chemotherapy	0992
Dentistry	0567
Education	0350
Hospital Management	0769
Human Development	0758
Immunology	0982
Medicine and Surgery	0564
Mental Health	0347
Nursing	0569
Nutrition	0570
Obstetrics and Gynecology	0380
Occupational Health and Therapy	0354
Ophthalmology	0381
Pathology	0571
Pharmacology	0419
Pharmacy	0572
Physical Therapy	0382
Public Health	0573
Radiology	0574
Recreation	0575
Speech Pathology	0460
Toxicology	0383
Home Economics	0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry	
General	0485
Agricultural	0749
Analytical	0486
Biochemistry	0487
Inorganic	0488
Nuclear	0738
Organic	0490
Pharmaceutical	0491
Physical	0494
Polymer	0495
Radiation	0754
Mathematics	0405
Physics	
General	0605
Acoustics	0986
Astronomy and Astrophysics	0606
Atmospheric Science	0608
Atomic	0748
Electronics and Electricity	0607
Elementary Particles and High Energy	0798
Fluid and Plasma	0759
Molecular	0609
Nuclear	0610
Optics	0752
Radiation	0756
Solid State	0611
Statistics	0463

Applied Sciences

Applied Mechanics	0346
Computer Science	0984
Engineering	
General	0537
Aerospace	0538
Agricultural	0539
Automotive	0540
Biomedical	0541
Chemical	0542
Civil	0543
Electronics and Electrical	0544
Heat and Thermodynamics	0348
Hydraulic	0545
Industrial	0546
Marine	0547
Materials Science	0794
Mechanical	0548
Metallurgy	0743
Mining	0551
Nuclear	0552
Packaging	0549
Petroleum	0765
Sanitary and Municipal	0554
System Science	0790
Geotechnology	0428
Operations Research	0796
Plastics Technology	0795
Textile Technology	0994

PSYCHOLOGY

General	0621
Behavioral	0384
Clinical	0622
Developmental	0620
Experimental	0623
Industrial	0624
Personality	0625
Physiological	0989
Psychobiology	0349
Psychometrics	0632
Social	0451

ABSTRACT

A Neural Network Model for Resource Leveling

Daniela Savin

A new neural network model aimed at solving the resource leveling (RL) problem in construction is developed. The model is derived by mapping a formulation of the RL problem as a quadratic augmented Lagrangian multiplier (QALM) optimization, onto an artificial neural network (ANN) architecture employing a Hopfield network. Specifically, it is shown that the augmented Lagrangian associated to the RL problem can be interpreted as the energy function of the Hopfield neural network (NN). The ANN architecture consists of two main blocks. The first is the Hopfield NN block, and the second is a control block for the adjustment of Lagrange multipliers in the QALM optimization. The latter is also used for the computation of the new set of weights of the Hopfield block.

A new methodology for the derivation of the weight-matrix of the Hopfield-based NN architecture for RL is also proposed. An in-depth study of the matrices generated from the formulation of the RL problem as a QALM optimization, has revealed some very useful structural properties. It is shown that due to these properties, it is possible to develop a flexible and computationally efficient procedure for the formulation and updating of the weight-matrix of the ANN configuration. This procedure allows for the on-line computation of the new set of weights of the Hopfield NN block with the adjustment of Lagrange parameters. The procedure also solves the key issue of mapping the QALM optimization formulation of the RL problem, onto an ANN architecture.

Finally, an experimental validation of the proposed model is presented along with its results.

ACKNOWLEDGEMENTS

I would like to express my gratitude and appreciation to my supervisor, Dr. Sabah Alkass, for initiating this research and for his guidance, support and encouragement throughout this study.

I am thankful to Dr. Paul Fazio for his helpful discussions and his partial financial support (FCAR équipe).

I am also grateful to Dr. Osama Moselhi for his valuable suggestions which contributed to my research.

I extend my thanks to the faculty and staff at the Centre for Building Studies for their help, encouragement and patience.

I owe my sincere gratitude to my family, especially my husband, Emanuel, and my son, George, for the wealth of encouragement and happiness they have brought to my life, and for having supported me during difficult periods of my research.

Many thanks are due to my close relatives and friends, for their help, advice and encouragement.

This work was partially supported by a FCAR grant given to Dr. S. Alkass, and by a Concordia University Graduate Fellowship, which are gratefully acknowledged.

TABLE OF CONTENTS

List of Figures

List of Abbreviations and Symbols

$\mathbf{A}$	Constraint-matrix
AI	Artificial intelligence
AL	Augmented Lagrangian
ALM	Augmented Lagrangian multiplier method
ANN	Artificial neural network
AoA	Activity-on-arrow network
AoN	Activity-on-node network
CPM	Critical path method
d_j	Duration of activity j
EFT_j	Earliest finish time of activity j
ESS	Early start schedule
EST_j	Earliest start time of activity j
ET	Earliest event time
$E(\mathbf{v})$	Energy function
$f(\mathbf{v})$	Cost (objective) function
$\mathbf{i}$	Vector containing the external inputs to each neuron
I	Total number of precedence constraints in a project
IEEE	Institute of electrical and electronics engineers
J	Total number of activities in a project
J	Total number of noncritical activities in a project
K	Project duration
LFT_j	Latest finish time of activity j
LT	Latest event time
$L(\mathbf{v})$	Augmented Lagrangian
NN	Neural network
NP-complete	Nondeterministic polynomial time complete

PDM	Precedence diagram method
PE	Processing element (neuron)
PERT	Program evaluation and review technique
QALM	Quadratic augmented Lagrangian multiplier method
r_j	Resource rate of activity j
RL	Resource leveling
TF_j	Total float of activity j
$\mathbf{t}$	Threshold vector
$\mathbf{v}$	Output (design) vector
$\mathbf{V}$	Non-singular decomposition of the constraint-matrix $\mathbf{A}$
VLSI	Very large scale integration
$\mathbf{W}$	Weight (connectivity) matrix
$\alpha_j, \beta_j, \gamma_j$	Penalty parameters
ϵ	Network error
κ	Extra penalty parameter
λ	Gain parameter
λ_j, μ_j, ν_j	Lagrange multipliers

Chapter 1

Introduction

1.1 General

A construction project can be viewed as a collection of activities performed in a logical sequence. In performing these activities, resources, which include equipment, technology, people, materials, time and money are used. Planning and control of the resources required to complete a project on time and within the budget, while meeting established specifications, are among the most challenging and difficult management responsabilities, which necessitate the utilization of construction management techniques.

Construction management is an ample and complex task. It requires knowledge of the construction processes, of modern management science, of decision support systems, as well as the use of computers (Hendrickson and Au 1989). Generally, construction management is differentiated from project management by the emphasis on the construction phase of a new facility (Kavanagh 1978). By contrast, project management embraces a broader perspective, including all the management activities related to the design, construction, operation and demolition of the facility. The main components of construction management are planning, scheduling and controlling project activities.

Construction planning comprises the identification of work tasks and the logical interdependencies among them. It constitutes the necessary precursor to scheduling. Scheduling involves the establishment of activities durations, project completion time, floats, and the identification of critical activities. It also involves the management of resources. Project control covers monitoring the work in progress, taking corrective actions, and updating the plan.

To succesfully manage projects, it is necessary for a management team to establish a plan for the utilization of the available resources, whether they are labor, materials, equipment or money. Resource management is one of the most important elements for planning, competitiveness and profitability in any construction project. It includes resource-constrained scheduling, time-constrained scheduling (resource leveling) and time-cost tradeoff. A basic distinction exists between resource-constrained and time-constrained scheduling. For resource-constrained scheduling, the focus is on utilizing limited resources in an effective manner. The scheduling objective is to extend the project duration, if needed, as little as possible beyond the original completion time, while the resource constraints are met. For time-constrained scheduling there are no resource limits, and the emphasis is on keeping the original project duration fixed.

Resource leveling (RL) is essential to the efficient utilization of resources and cost control of a project. Network techniques usually produce uneven resource requirement profiles, whenever there is an assumption of unlimited resources. These resource profiles generate high costs, uneven cash flows, and generally reflect inefficient use of existing resources.

A review of the literature reveals that the major research emphasis has been on the development of resource-constrained scheduling procedures. However, Burgess and Killebrew (1962) recommended that resource leveling should routinely precede any resource constraint considerations. Once the leveling has been completed, a maximum resource requirement would have been established. If the maximum required is greater than the available resources, alternative actions need to be considered to manage the peak resource demand periods, including the possiblity of extending the project duration. However, when the project duration is extended, the leveling process should be repeated.

Supporting Burgess's position on the importance of resource leveling, Fleming et al. (1987) reported the view of the resource leveling supporters who indicate that a "schedule should never be approved until resources have been initially allocated to the tasks being scheduled, analyzed and then leveled to their most efficient use".

The project management literature has focused on two primary approaches with regard to resource leveling: optimization and heuristics. Optimization techniques seek the best (optimum) solution, while heuristic techniques produce good near-optimal solutions.

Different formulations of the RL problem as constrained optimizations have been proposed and discussed in the literature (Easa 1989; Ramlogan and Goulter 1989; Movassaghi and Beidoun 1988; Karaa and Nasr 1986). However, due to the combinatorial complexity of the RL problem, the optimization methods offer computationally-efficient solutions only for small to medium-sized projects (Antill and Woodhead 1990; Moder et al. 1983; Ahuja 1976). For larger projects, heuristic techniques have been developed to solve the RL problem (Seibert and Evans 1991; Harris 1990; Woodworth and Willie 1976). The ability of handling larger projects is achieved by relaxing the goal of searching for an optimal solution. Specifically, a heuristic technique is concerned only with achieving lower values for the objective function associated with an optimization problem, while satisfying the network constraints. In order to arrive at a near-optimal solution for a RL problem, it would be necessary to employ different heuristic methods in parallel (Ahuja et al. 1994). Therefore, the investigation in the development of alternative heuristical methods for solving the RL problem is still very intensive, leading the way to the application of more effective techniques, such as neural networks.

Neural networks are increasingly applied to civil and construction engineering. (Flood and Kartam 1994; Savin et al. 1994; Garrett 1992; Moselhi et al. 1991). It is to be noted that the most utilized architecture has been the backpropagation. Very little work has been done using other kinds of architectures, specifically the Hopfield architecture (Savin et al. 1995; Kobayashi and Nonaka 1990; Gulati et al. 1987). The recent and promising results in solving combinatorial optimization problems by using neural networks (NNs) (Peterson and Anderson 1988), have encouraged the beginnings of research in the development of heuristic methods for solving the RL problem using Artificial Neural Network (ANN) models (Shimazaki et al. 1991). This research investigates the potential use of ANN techniques for RL.

1.2 Motivation

The motivation for studying resource leveling problem arises from the following:

- Current techniques (optimization, heuristic) are not sufficient in solving the resource leveling problem.
- Resource leveling is a combinatorial NP-complete (nondeterministic polynomial time complete) problem. For NP-complete problems, no algorithm capable of providing an exact solution to the problem, in a computational time which is a polynomial in the size of the problem, is yet known. For resource leveling, as the number of noncritical activities increases, the required number of all possible combinations of activities within their available floats would be impractical. Therefore, there is a need for a procedure to overcome this combinatorial complexity. Artificial neural networks have been found to be a new promissing approach to solve such problems efficiently and in a short period of time.

1.3 Scope and Objectives

The scope of this work is limited to time-constrained scheduling using artificial neural networks. The fluctuations in the pattern of resource usage are minimized by shifting the noncritical activities within their available floats, while maintaining the original project duration unchanged.
The objectives of this research are as follows:

- To explore the current techniques used for resource leveling and identify their shortcomings.
- To introduce a procedure to overcome the shortcomings of the resource leveling techniques.
- To test the potential use of neural networks to reduce the complexity of resource leveling, based on high degree of interconnectivity and parallel processing.
- To develop a neural network model for resource leveling.

1.4 Methodology

In order to achieve the above mentioned objectives, the following tasks were performed:

- Extensive literature review on both, the present resource leveling techniques and the neural networks applications to scheduling.
- Formulation of the resource leveling as an equality-constrained optimization problem.
- Formulation of the resource leveling problem as an augmented Lagrangian.

- Development of a neural network model for resource leveling, by mapping the augmented Lagrangian onto a Hopfield neural network.
- Development of a flexible and computationally-efficient procedure for the formulation and updating of the weight-matrix of the Hopfield network.
- Experimental verification of the proposed artificial neural network model for small-sized construction projects.

1.5 Organization of the Thesis

The thesis is organized as follows: In Chapter 2 a review of the resource leveling theories and the relevant background material is presented. Different approaches to resource leveling discussed in the literature are described and their shortcomings identified. This is followed by a brief description of Artificial Neural Networks in Chapter 3 with emphasis placed on using Hopfield NN for optimization problems. Chapter 4 presents a summary of the Augmented Lagrangian Method, followed by Chapter 5, which includes a detailed description of the proposed neural network for resource leveling. Chapter 6 covers a description of a procedure for the formulation and the on-line updating of the weight-matrix of the Hopfield network. An experimental verification of the proposed artificial neural network model, along with its results are presented in Chapter 7. Finally, Chapter 8 summarizes the work done during this research and concludes its findings, together with proposed future work.

Chapter 2

A Review of Resource Leveling

2.1 Project Scheduling

2.1.1 Introduction

Project scheduling establishes the completion time of the project, when the activities may be performed and with what resources. The scheduling process integrates information on several aspects of the project, including the estimated durations of activities, constraints imposed by the availability of resources, due-date requirements, and the technological precedence relations among activities established during the planning process. This information is processed into a schedule, which provides an essential communication and coordination link between the individuals and organizations participating in constructing the project. Schedules are the primary working tools for project planning, evaluation, and control.

Bar charts and project-networks are two scheduling techniques mainly used in the construction industry. Bar charts, also called Gantt charts, remain one of the most widely used project scheduling methods. They are simple to generate, easy to understand and use. The bar chart is a graphic presentation, in the form of bars, which shows the anticipated start and finish dates of each activity in a project. The major limitation of bar charts is their inability to show enough interdependencies among activities and time-resource trade-offs (Harris and McCaffer 1989).

Unlike the bar charts, networks graphically portray the relationships between the activities and milestones in the project. Several techniques have evolved in the late 1950s for organizing and representing this basic information. Best known today are PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method). The major diference between the two is that CPM assumes that activity duration is deterministic, while PERT views the time to complete an activity as a random variable that can be characterized by an optimistic, a pessimistic, and a most likely estimate of its duration (Moder et al. 1983).

Over the years, a large number of variants has arisen to address specific problems such as complex activity interdependencies (Precedence Diagram Method - PDM), project cost control (PERT/COST), time-cost tradeoff, and the multitude of uncertainties found in the construction environment (Graphical Evaluation and Review Technique - GERT, Monte Carlo simulations) (Antill and Woodhead 1990; Moder et al. 1983). A special technique, the Line of Balance (LOB), has been developed for scheduling repetitive projects (Harris and McCaffer 1989), and also efforts have focused on the development of more performant scheduling techniques for projects comprising both repetitive and non-repetitive activities (Moselhi and El-Rayes 1993; O'Brien, J. 1975).

Considerable research works have been carried out in recent years on the use of artificial intelligence (AI) techniques for project planning and scheduling (Ahuja et al. 1994), such as knowledge-based expert systems (KBES) (Moselhi and Nicholas 1990; Echeverry et al. 1989; Navinchandra et al. 1988; Hendrickson et al. 1987; Levitt and Kunz 1985), including fuzzy sets (Lorterapong 1995; Chang et al. 1990), and Artificial Neural Networks (ANNs) (Mawdesley and Carr 1993), which have significantly changed the features of the traditional scheduling process. The integration of network-based and AI-based techniques will be a good advancement in project planning and scheduling, as they complement each other by offering a problem solving strategy and a modeling system to describe the project plans (Morad and Vorster 1993; Kartam et al. 1993).

In order to demonstrate the possible use of ANNs in time-constrained scheduling, it is useful to briefly describe the project scheduling techniques, such as CPM.

2.1.2 Critical Path Method

The critical path method " ... is the representation of a project plan by a schematic diagram or network that depicts the sequence and interrelation of all the component parts of the project, and the logical analysis and manipulation of this network in determining the best overall program of operation" (Antill and Woodhead 1990). The diagram that represents the project consists of a network of arrows and nodes. The two most popular approaches are either to place the activities on the arrows (AoA) and have the nodes signify milestones, or to place activities on the nodes (AoN or Precedence Diagram Method (PDM)), and let arrows show precedence relations among activities (Ahuja et al. 1994). This research work focuses on using activity-on-arrow (AoA) networks, therefore the subsequent exposition will refer only to this class of networks.

In constructing an AoA network, an arrow is used to represent an activity, with its head indicating the direction of progress of the project. The precedence relations among activities are introduced by defining events. An event represents a point in time that signifies the completion of one or more activities and the beginning of new ones. The beginning and ending points of an activity are thus described by two events, known as the head and the tail. Activities originating from a certain event cannot start until the activities terminating at the same event have been completed. Figure 2.1(a) shows an example of a typical representation of an activity (i, j), with its tail event i and its head event j.

The time required for each activity needs to be estimated; this estimate of the

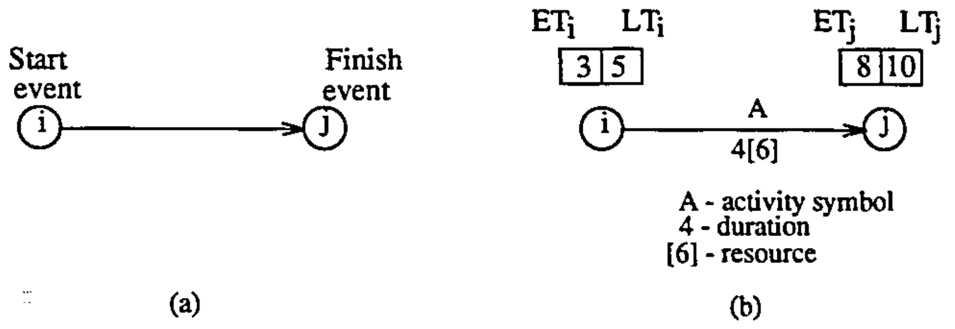


Figure 2.1: Typical representation of an activity in activity-on-arrow networks

duration is based on knowledge of the work, experience or historical data. Once estimated, the duration of each activity is marked next to the corresponding arrow in the network logic, as it can be seen in Figure 2.1(b). The earliest possible time of each event is then calculated and written in the left-hand square alongside each event. The calculation of the earliest event times (ET) is known as the forward pass. The reverse process, the backward pass, determines the latest possible time for the event, that is the latest possible time for each activity to finish without delaying the completion date of the project. The latest event time (LT) is calculated and written in the right-hand square alongside each event, as it is illustrated in Figure 2.1(b). Note here that for a particular activity, the earliest event time (ET) of its beginning node represents the earliest start time (EST) of the activity, and the latest event time (LT) of its ending node represents the latest finish time (LFT) of that activity. The earliest finish time (EFT) and the latest start time (LST) of the activity are calculated as shown in Figure 2.2.

Having completed the forward and backward passes, the earliest and latest times of each event are known, as it is illustrated in Figure 2.3. From this, the total float, or spare time available for each activity, can be calculated. Total float is the total amount of time by which the activity could be extended, or delayed, without affecting the project completion date. Total float is the total time available for the

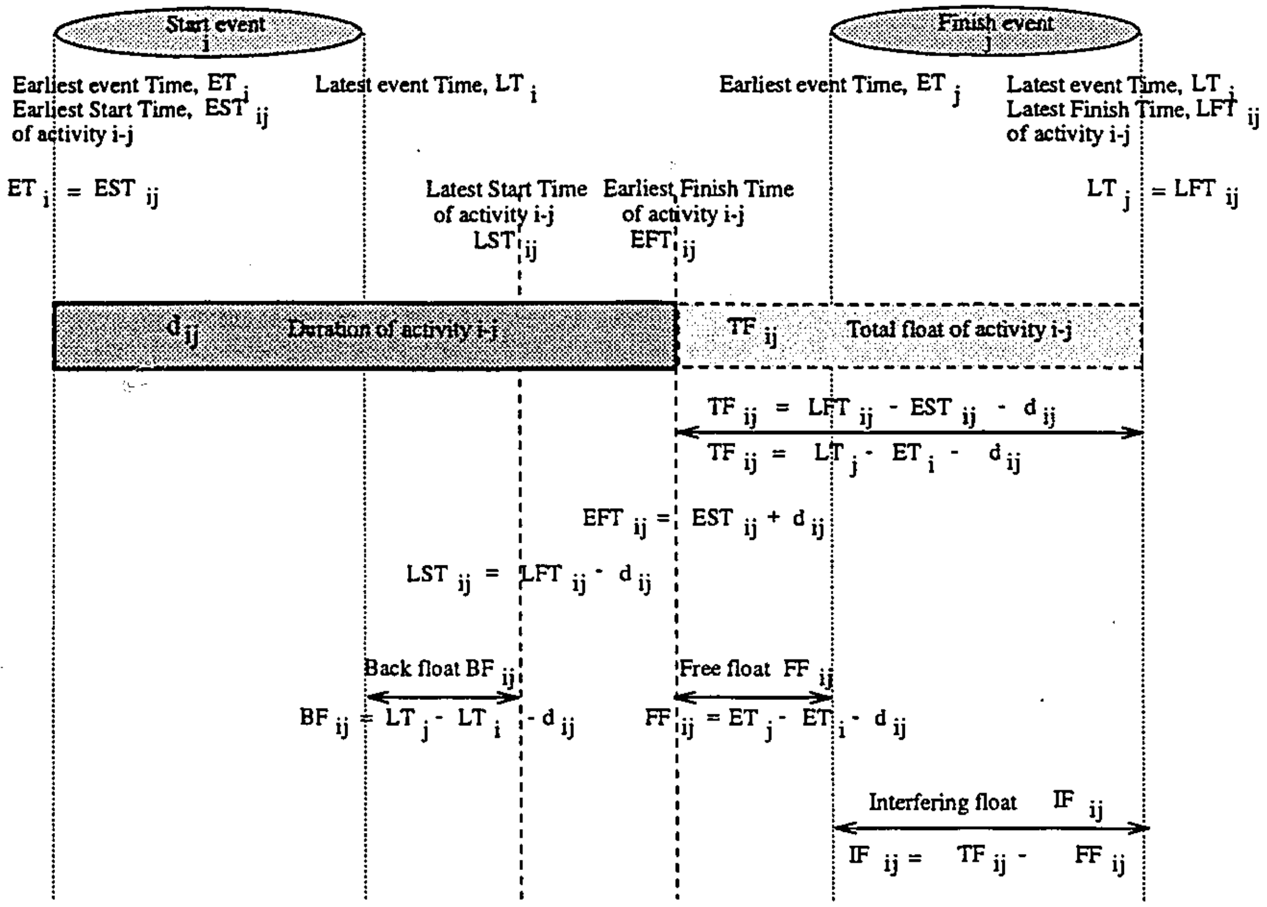

Start event i
Finish event j
Earliest event Time, ET i
Earliest Start Time, EST ij of activity i-j
Latest event Time, LT i
Earliest event Time, ET j
Latest event Time, LT j
Latest Finish Time, LFT ij of activity i-j
ET i = EST ij
Latest Start Time of activity i-j LST ij
Earliest Finish Time of activity i-j EFT ij
LT j = LFT ij
d ij Duration of activity i-j
TF ij Total float of activity i-j
TF ij = LFT ij - EST ij - d ij
TF ij = LT j - ET i - d ij
EFT ij = EST ij + d ij
LST ij = LFT ij - d ij
Back float BF ij
BF ij = LT j - LT i - d ij
Free float FF ij
FF ij = ET j - ET i - d ij
Interfering float IF ij
IF ij = TF ij - FF ij

Figure 2.2: Illustration of the activity start and finish times and the float

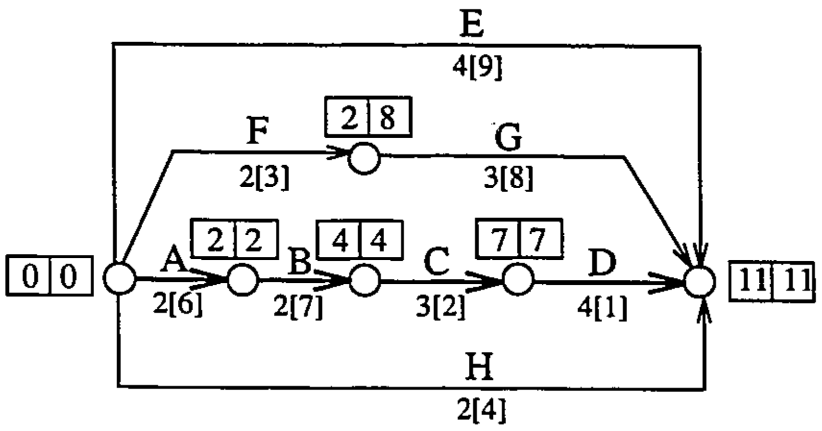
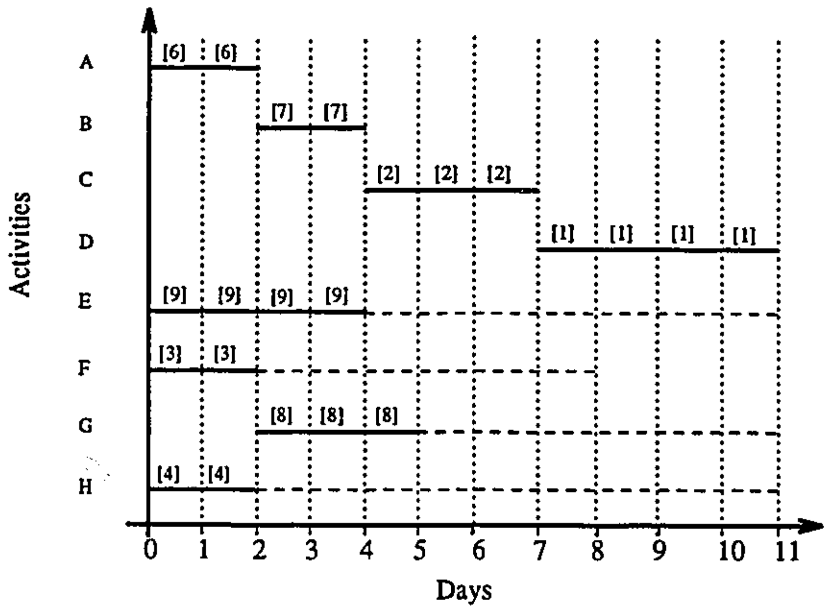


Figure 2.3: Example of an activity-on-arrow network

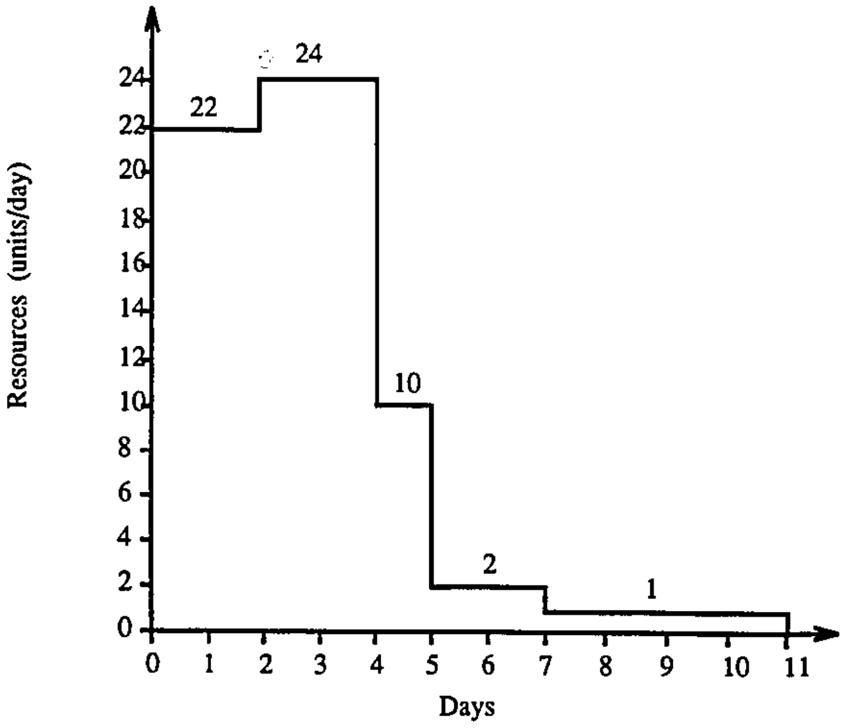
activity less its duration, i.e., the latest time of finish event less the earliest time of the start event less the duration. Activities with a total float equal to zero are critical activities. The sequence of critical activities connecting the start and end points of the project is known as the critical path, which is the longest path in the network. The length of the critical path represents the shortest possible time to complete the project. More details about CPM calculations can be found in (Ahuja et al. 1994; Moder et al. 1983; Lester 1982).

Based on CPM computations, a bar chart is then created (Ahuja et al. 1994). This bar chart (which is a version of the Gantt bar chart) is still a very popular scheduling tool. It enumerates the activities to be performed on the vertical axis, and their corresponding durations on the horizontal axis. On a CPM bar chart an activity is represented by a continuous line, whereas its float is illustrated by a dashed line, as it is shown in Figure 2.4(a). It is possible to schedule activities by either early start or late start logic, which means the float could precede the activity or succeed it. In the examples presented in this thesis, all activities begin at their earliest start times, therefore the floats succeed the activities.

In estimating the duration of an activity, the resources required to perform that activity have to be considered. The resources can be written alongside each



(a) Bar chart



(b) Resource histogram

Figure 2.4: Initial allocation of resources, before leveling

arrow in the network, next to activity duartion, in square brackets, as it is shown in Figure 2.1(b). The CPM method assumes that the only constraints on the schedule are precedence relations among activities, without taking into consideration the role of resources in establishing an efficient schedule. This is not a realistic approach, since the allocation of resources may cause the original schedule to be unfeasible. Thus, the project planner must be concerned not only with the logic and time constraints of the activities in a project, but also with the management of the resources.

Resource management is the process by which the planner of the project decides which resources to obtain, from what source, when to obtain them, and how to use them. Techniques used in resource management have received increasing attention in recent years, leading to the development of new procedures. There are two important aspects of resource management: resource leveling (time-constrained scheduling), in which the project must be completed by a specific date, and resource allocation (resource-constrained scheduling), in which the project must be completed with the limited resources available, even if it means extending the project deadline.

2.1.3 Resource Leveling

Time-constrained scheduling, also known as resource leveling, is considered to be an important problem facing construction schedulers. It affects project productivity and costs. The main objective of RL is to minimize the fluctuations in the pattern of resource usage, making the resource requirements as uniform as possible. This is accomplished by using an initial CPM schedule, and shifting the noncritical activities within their available floats, while maintaining the original duration intact.

Large variations in resource requirements result in additional costs associated with:

- Obtaining and releasing resources to meet the varying resource requirements

- Productivity and perhaps quality losses due to learning curve effects
- Idle resources during periods of low resource requirements
- Overtime costs for peak resource requirement periods
- Management time to coordinate the resources
- Acquisition and storage due to wide fluctuations in material usage
- Difficulty of coordinating and controlling large numbers of people during periods of peak manpower usage
- Uneven cash flows

Therefore, the project team must minimize the fluctuations in the resource requirement profile, since a more steady usage rate leads to lower resource costs.

Based on the CPM bar chart, by summing the resources required of the activities which occur on any given unit of time (e.g., day, week), the required project resources can be represented as a function of time. The graph of resource requirement as a function of time is called a profile, or the histogram of required resources. For example, consider the hypothetical construction project which is modelled by the arrow notation CPM network, shown in Figure 2.3. The project duration is 11 days, and the critical path runs through activities A, B, C, and D. In addition to the duration, a resource requirement (in square brackets) for each activity is also shown, assuming that only one type of resource is used. The bar chart for the early start schedule is shown in Figure 2.4(a). The corresponding resource requirement profile is depicted in Figure 2.4(b). As it can be seen, the early start schedule generates a widely varying profile.

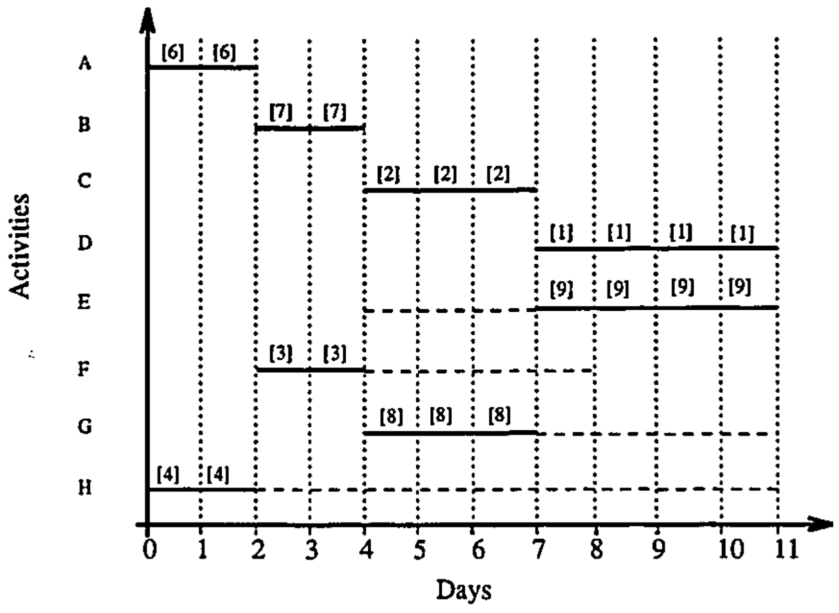
The fact that the amount of resource required varies with time may result in lower productivity, added project cost, or inability to profitably obtain the resources in such a fashion. Therefore, there is a need to smooth, or to level the peaks and valleys in the resource profile. Using float is one way to reshape resource requirements. It may be possible to achieve higher resource utilization and lower costs by exploring different assignment patterns, since it is always possible to start an activity within the range defined by its early and late schedules.Based on this consideration, a cycle of schedule updates may become necessary, until the schedule corresponding to the lowest cost is obtained. Generally speaking, the smoother the resource profile, the lower is the overall cost (Ahuja et al. 1994). For the example-project considered in Figure 2.3, by rescheduling the noncritical activities E, F and G within their available floats, as it is shown in the bar chart of Figure 2.5(a), a leveled (smoothed) resource histogram is obtained, as illustrated in Figure 2.5(b). The fluctuations in the pattern of resource usage from Figure 2.4(b) have been reduced, and the profile of required resources after leveling is constant, as it can be seen in Figure 2.5(b).

In a case when the final profile of resource usage exceeds the maximum amount of resources available, then the use of the previously mentioned resource-constrained scheduling techniques, is required.

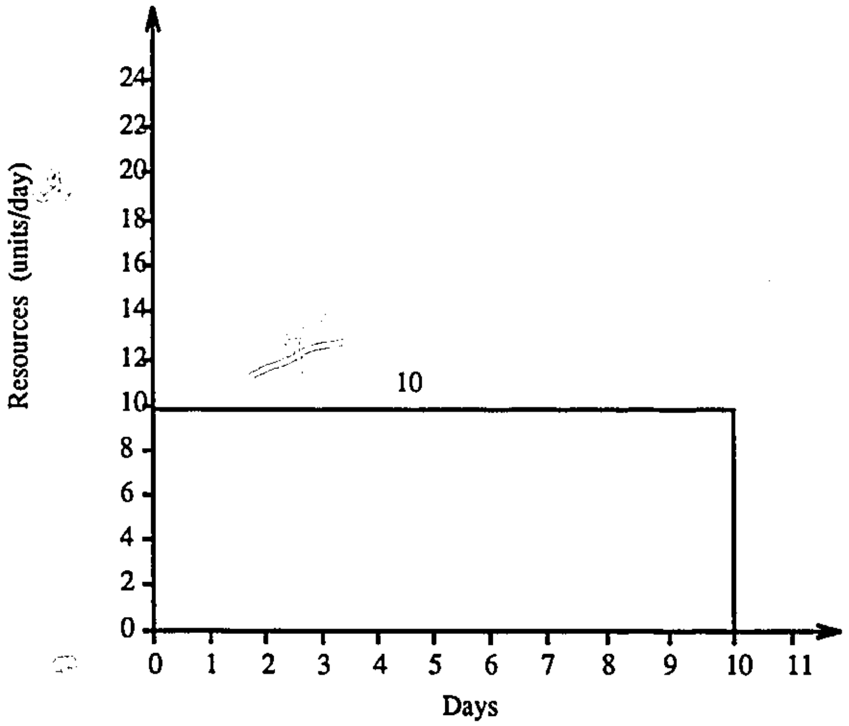
2.1.4 Resource-Constrained Scheduling

Resource-constrained scheduling techniques are designed to produce schedules with limited available resources , which causes the project duration to extend beyond the original critical path length (Ahuja et al. 1994; Moder et al. 1983; Adrian 1973). The construction of a resource-limited profile is similar to that of the unlimited resource approach, except that if the total resource demand of an activity exceeds the specified limit, then that activity must be dalayed outside its time-span. When this happens, project delays are inevitable, unless corrective actior can be taken immediately.

Resource-constrained scheduling is an important issue in project scheduling, and has been a major concern of research since the introduction of CPM/PERT



(a) Bar chart



(b) Resource histogram

Figure 2.5: Final allocation of resources, after leveling

network scheduling techniques. A number of techniques for resource-constrained scheduling have been developed through the years (Karshenas and Haber 1990; Allam 1988; Patterson 1984; Kurtulus and Davis 1982; Talbot and Patterson 1979; Panwalkar and Islander 1977; Davis and Patterson 1975). A new and promissing technique, employing fuzzy set theory for modeling the uncertainties associated with the durations of project activities and the resource availabilities, was recently proposed by Moselhi and Lorterapong (Lorterapong 1995; Moselhi and Lorterapong 1993).

Resource-constrained scheduling is a complex, challenging and interesting problem. However, the scope of this thesis is limited to time-constrained scheduling. Therefore, the most important findings from the literature, related only to resource leveling, are reported next.

2.2 Resource Leveling Techniques

Unlike resource allocation techniques which use the project duration as the measure of their effectiveness, resource leveling techniques do not have such a measure of effectiveness, making them difficult to assess (Seibert and Evans 1991), i.e., for given resources, a shorter project duration reflects a better allocation of those resources. The sum of the absolute daily deviations in resource usage may find the best solution in some cases. The sum of squares of deviations may render the best schedule in other cases. As a result, no universally-accepted criterion exists for determining the best solution for a resource leveling problem (Shah et al. 1993). The simplest measure of the efficiency of a leveled schedule would be a constant amount of each resource for the project duration. However, different types of resources are needed in varying amounts over the life of the project (Adrian 1973). Most project-scheduling software packages require the user to input an initial resource curve for each type of resource, prior to leveling the resources. Resource curves have been discussed in few articles found in the literature (Seibert and Evans 1991; Easa 89; Leachman 1983; Galbreath 1965); most of them assume that resource utilization rate is constant, and the optimum schedule after leveling should have a rectangular distribution of resources. How well the resources are leveled versus the assumed initial profile is a question that many researchers have tried to address; The resource improvement coefficient (Martinez and Ioannou 1993) resource-utilization factor and the sum of the residuals squared (Seibert and Evans 1991), the minimum moment value (Harris 1978), represent only few examples of the methods reported in the literature.

Two main techniques for resource leveling have been reported in the literature: optimization and heuristic. Optimization techniques aim at producing the best leveled schedules. Heuristics are approximate techniques which provide acceptable, but not necessarily "the best" solutions in most cases. The major research emphasis has been on the development of heuristic-based rather than optimization-based techniques. Heuristics provide good, practical solutions in most cases; but as resources become more scarce and the competition among project activities stronger, optimal solutions will be increasingly sought by project managers (Shah et al. 1993). Optimization techniques draw optimal solutions; but the computational capacity required by them often exceeds the present hardware capacities of even the mainframe computers, especially for large projects that are common in the construction industry. However, with the advent of AI procedures, efficient algorithms and high-performance computers, optimization techniques for resource leveling are expected to become feasible.

2.2.1 Optimization Techniques

Optimization techniques for RL require a mathematical formulation of the problem. In general, the RL is formulated as an integer or mixed integer linear programming problem, in which an objective function has to be minimized subject to a set of constraints. Most of the researchers have used comercially available optimization software packages, e.g. LINDO, and very few others have tried enumerative algorithms (i.e., implicit enumeration and branch-and-bound). A survey of the general purpose integer linear programming algorithms can be found in Rao (1984) and in Geoffrion and Marsten (1972). Only one dynamic programming approach to resource leveling was proposed by Petrovic, in 1968. The method uses the sum of the squared deviations from the mean resource requirement as the dependent criterion. Although Petrovic claims that his approach is capable of handling the multiple resource problem, his description and formulation of the problem is limited to the single resource problem. The author states that the "... high dimensions of the functional equations make the solution difficult, and may become, in practice, serious limitations to the feasibility of the general computation".

A mathematical model that uses the complete enumeration of all possible combinations of shifting the noncritical activities within their allowable floats, has been presented by Ahuja (1976). This author showed that the comparison of sum of squares of the resource requirement for each time unit does help select the most leveled schedule, but the comparison leads to a rectangular distribution. Therefore, in order to produce a more gradual distribution (e.g., parabolic), it is necessary to apply another criterion. The criterion used by the model is the minimization of the fluctuations of day-to-day resource changes. A major drawback of this method is that its application becomes less feasible as the size of the network increases; the number of all the combinations, that have to be investigated to arrive at the final solution, exceeds the computational capacity of even the most modern and large computers.

In 1986, Karaa and Nasr developed an optimization model that minimizes the total cost of leasing additional resources in construction, under the constraint of maximum and most efficient use of owned equipment and contracted labor force. The model has a mixed integer-linear programming structure, and derives the schedule for equipment rentals, as well as the utilization scheme for owned equipment and other available resources. The model can be used as an estimating tool for multi-project, multi-resource planning and sharing, and as a means to implement the most efficient utilization of resources throughout the duration of a project. The model has the capability of mapping continuous, intermitent, uniform and nonuniform activities. The main limitation of the model lies within its dimensionality. As the project increases in complexity, the number of integer variables becomes too large to be handled by existing linear programming software packages (e.g., LINDO). The authors suggested a way to reduce the dimensionality problem, by decomposing the network in subnetworks of activities between two major milestones.

A mixed integer programming model, that minimizes the sum of the squares of deviations from the mean resource requirement, was introduced by Movassaghi and Beidoun (1988). This optimization algorithm for RL assumes a single resource requirement, and a constant rate of resource demand. The objective function is subjected to a set of constraints relating the available total float times to the precedence relationships of the affected activities. In order to solve their mixed integer programming problem, the authors have used the branch-and-bound algorithm and the dual simplex method. The existence of a large number of constraints and variables is the main limitation of the procedure. The authors stated that the computer time increases almost exponentially with the number of integer variables.

Easa (1989) proposed an integer-linear optimization model for resource leveling for a single resource and continuous activities. The objective function of the model minimizes the absolute deviations between the resource requirements and a uniform resource level, between consecutive resource requirements, or between the resource requirements and desirable nonuniform resource levels. Extensions of the model to multiple resources and trade-off of cost scheduling are suggested. The model is applicable to AoA or AoN networks, and is intended for small to medium sized construction projects. The model requires as the input the CPM scheduling results, and the resource rates of activities. This information is used by an interface program which establishes automatically a file containing the objective function and the constraints of the model. This file is then input to an integer-linear optimization program, which produces the optimal solution. As is the case with integer optimization models, this model has also the dimensionality limitation. For large projects, the number of binary variables and constraints may be too large to be handled by existing computer packages.

Another mixed integer model for RL was developed by Ramlogan and Goulter (1989). The model schedules activities within their available free floats, such that the project duration, as calculated by the CPM, does not increase. However, the estimated activities durations used in the critical path analysis are not considered the model's constraints. The scheduled activities durations are determined by the model. This procedure has three objectives, all related to the overall objective of total resource leveling: 1) resource leveling for the individual activities, 2) minimization of total duration of the individual activities (in essence making the activity occur on consecutive days), and 3) overall resource leveling of the project. These objectives are placed within the formulation in a weighted multi-objective framework. The functionality of the model is demonstrated by application to an example project, using LINDO optimization package. The model is complex, and would "theoretically" give an optimal project schedule. The computational requirements of the model are quite intensive. However, the authors suggested that when more efficient branch-and-bound or other integer algorithms are developed, the efficiency will improve significantly.

Shah, Farid and Baugh (1993) proposed a Scheme language program, interfaced with LINDO and S-plot, to find the minimum resource limit required to finish a project within its planned duration. Multiple resources can be scheduled by transforming various resource requirements into a common base, or by using the weighting method. The computer program uses an integer-linear programming model, originally developed by Wiest (1977) for resource allocation, and adapted by the authors for resource leveling. Instead of requiring a resource limit as input, the program calculates the minimum resource limit required to finish the project within its planned duration. By minimizing the required resource limit, the resource histogram is automatically leveled. Under this resource limit, however, a better solution may yield either a lower sum of absolute deviations, or a lower sum of squares of deviations. Therefore, generally the program provides a near-optimal solution, and it is applicable only to small projects.

Resource leveling is a NP-complete problem, which usually takes time of $exp(n)$ order,and therefore, when the size of the problem increases it becomes intractable. Typically, if the combinatorial optimization problem is of size n, then the possible solutions are of the e^n or $n!$. For NP-complete problems, no algorithm is known which provides an exact solution to the problem, in a computational time which is a polynomial in the size of the problem. As the number of noncritical activities increases, the required number of all possible combinations of activities within their available floats would be impractical. For example, if a project has 10 noncritical activities, each having a float of 6 time units, there will be 6^{10} (over 60 million) possible combinations (Easa 1989). Therefore, there is a need for a procedure to overcome this combinatorial complexity.

2.2.2 Heuristic Techniques

Heuristic techniques produce good feasible schedules using rules of thumb to determine priorities among the competing activities. Some of the predominant priority rules commonly used are: the least float, the earliest start time, the minimum late start time, the minimum early finish time, the minimum late finish time, the largest duration and the shortest duration (Ahuja et al. 1994, Antill and Woodhead 1990, Moder et al. 1983, Harris 1978). Heuristic rules may be employed using either parallel or series methods. The series method accomplishes leveling by allocating resources to activities in series (one activity at a time, from start to finish). The parallel method allocates resources to activities, one day at a time (Ahuja et al. 1994, Antill and Woodhead 1990, Moder et al. 1983, Harris 1978). The best leveling may be obtained by trial and error, applying different heuristics for the same problem and choosing the best alternative. Heuristic techniques can handle large projects, but the solution they provide might not be the optimum one.

The simplest and the most popular of the algorithms reviewed, was developed by Burgess and Killebrew (1962). It is a systematic heuristic procedure for leveling a single resource. They found out that the sum of squares of the period by period resource requirements decreases as the peaks and valleys in the profile are leveled, and this sum of squares reaches a minimum for a schedule in which the period to period resource requirement is constant. The procedure begins with all activities scheduled at their earliest possible starting times. Beginning with the last activity in the network, activities are shifted one period at a time to later starting times, until their total float is exhausted. For each alternative, the position of the activity and the resulting sum of squares of the resource profile are evaluated. The activity is then rescheduled in the position that yields the lowest sum of squares. The algorithm's inability to guarantee the optimum schedule, with respect to minimizing the variance in the resource levels, is considered to be its major limitation. However, a procedure for an optimal solution would require that all feasible combinations of activity locations to be evaluated, which is computationally impractical for networks of significant size. A second limitation of this algorithm arises from the fact that it handles a single resource, when in practice projects typically utilize multiple resources.

Levy, Thompson and Wiest (1963) described a multiship (multiple project), multishop (multiple resource) leveling heuristic used to minimize peak manpower requirements for shops (resources) in a naval shipyard. An ample description of this heuristic technique, called MS^2 (for multiship, multishop), or simply Wiest's procedure, can also be found in Wiest and Levy (1977). The algorithm consists of two steps. In the first step, the workloads of all shops are smoothed, simultaneously decreasing the maximum manpower availability for all shops one unit at a time. When a peak load period is encountered, the algorithm randomly selects an activity from a list of activities possessing float greater than zero. This activity is then shifted some number of periods to the right of the peak period. The specific number of periods that the activity is shifted is randomly selected by the algorithm. When it is no longer feasible to further reduce the amount of the resources, the second step of the algorithm starts by performing further smoothing on individual shops. The resources are leveled sequentially, from the most expensive to the least expensive. A limitation of the sequential process is that the leveling of any given resource places additional restrictions on the potential starting times of the activities using the other resources. The final resource profiles are dependent on the order in which the resources are leveled. The evaluation criterion for the schedules produced by this algorithm is the resources' costs, and its objective is to minimize these costs. A major limitation of this algorithm arises from the heuristics used to select an activity to be shifted, and the number of periods that activity is to be shifted. Since selecting an activity and its shifting periods are randomly done, then the probability of producing a good solution is relatively small.

Galbreath (1965) proposed a computer program to handle multiple resource leveling, using heuristics. The program examines the total resource requirement for each resource type and the total for each day. If the resource requirement is greater than the available resources, then the activities scheduled on that day are shifted within their allowable free floats, or total floats (if free floats have been reduced to zero), until the resource requirement matches the availability. After having examined all the activities in this fashion, the extra resource required (if any) would be the new total resource requirement for that day. The next resource is then considered for the same day, in a similar fashion, until the last resource has been used for that day. This is repeated until the final day of the project has been reached. An important feature of the program is that it allows the activities to split into segments during the leveling process. Another advantage is that each resource availability level can be presented to any desired curve. One of the limitations of this method is associated with the order in which activities are selected for rescheduling; different selecting rules yield different leveling solutions. Relocating an activity without regard to the area into which it is placed is also another shortcoming of the algorithm. Relocating an activity strictly on the basis of an excess of requirement over availability for one resource considered along, is a scheme open to criticism, too.

Woodworth and Willie (1976) described a heuristic algorithm for the multiple project, multiple resource problem. Although the leveling algorithm is not discussed in detail, they described it as a modification of the Burgess and Killebrew procedure. They implemented an alternative heuristic method, which starts with the first activity in each project. On the second and subsequent passes through each project, the activities are moved to the left, to their earliest starting times, as well as to the right, to later starting times. Woodworth and Willie deal with the multiple resource problem by prioritizing the resources for analysis. The criterion used to evaluate the levelness of the produced schedules is the sum of squares for each individual resource, across all projects. As a heuristic algorithm, this procedure has the same limitations as discussed earlier for the other heuristic procedures. The limitation associated with sequential scheduling of different resource types, dicussed in the Levy, Thompson and Wiest algorithm, is also applicable to this one. Finally, the authors conclude that the algorithm has no trade-off functions , therefore the resulting schedule may not be the least-cost schedule.

In an attempt to extend and refine the efforts of the above mentioned authors (i.e., Antill and Woodhead 1990; Wiest and Levy 1977; Adrian 1973; Galbreath 1965; Burgess and Killebrew 1962), Harris (1978) developed the minimum moment algorithm. This heuristic method assumes that the activities are continuous, resource rates are constant, and the network logic is fixed. It can be used for AoA or

PDM networks. The minimum moment algorithm considers the resource histogram as an area, and sets its objective as the minimization of the moment of that area, along the x-axis. The method uses the concept of Burgess' sum of the squares of the daily resource sums, and develops an improvement factor to select the activity to be shifted. Shifting begins with activities on the latest sequence step, and the improvement factor is used to select the particular one on the step , to be shifted. When all the activities on a step have been considered, the next earlier step is examined. This process continues until the first step has been reached. Free float is used to set the range of possible shifting of activities. The order is then reversed, starting with the first sequence step and proceeding to the last step, using the back float to set the range of shifting. Harris suggests the leveling of multiple resources in series, as described by Antill and Woodhead (1990), or by combining them in a single application, by assigning weights.

Martinez and Ioannou (1993) described a computer program (CPMLevel) that utilizes a modified version of the minimum moment algorithm (Harris 1978), to smooth the resource histogram in both AoA and PDM networks. The authors state that, in addition to solving the CPM and resource leveling problem, CPMLevel also performs limited resource allocation, and uses graphics animation to show the effect of each step in the leveling procedure, of the resource histogram. In comparison with the minimum moment algorithm, CPMLevel provides the opportunity to use late sequence steps instead of early sequence steps, on either or both the forward and the backward passes. The authors demonstrated that dynamically assigned sequence steps can provide better results, when compared to assignment by early sequence steps.

Thesen (1978) designed a computer program for monthly development of detailed daily schedules of manpower assignments. The program was designed to handle large projects, multiple resources and "any number of different project networks". Although it is reported that this program is capable of leveling the manpower assignments, the description and formulation of the problem is clearly limited to scheduling under limited resources.

Leachman (1983) described a complex heuristic technique, for handling multiple resources, in multiple projects environment. The author emphasized the variation of the intensity of activity resource loading, as a tool for leveling the overall resource load profiles, in contrast to the traditional approach of simply rescheduling fixed intensity activities. Activity intensity variables are defined, which measure activity demand rates for resources and consequent activity durations for the production of each output unit. A heuristic approach consisting of an iterative nonlinear programming procedure was presented, which computed activity durations (intensities) for the minimization of resource capacity costs, subjected to meeting construction due dates. A major criticism of this method is the complexity of the algorithm. Ephremidis (1987) briefly described a computer program for the multiple project, multiple resource leveling problem. Although the leveling algorithm is not discussed in detail, the author describes it as a modification of the Wiest procedure (Wiest 1977). He also states that the program can deal with the problem of optimizing the project duration time with limited resources.

Harris (1990) introduced a new heuristic model for resource leveling, called PACK. The model assigns project activities to specific days, so that the final resource histogram approaches a rectangular shape, and its moment approaches a minimum value. This heuristic method adopts a new procedure, where a base resource histogram is determined by using only the critical activities and those that have total floats less than or equal to the activity duration. Then, each noncritical activity is directly positioned in time, and its resource contributions are added to the base histogram. The author describes three rules for determining the scheduling priority of the noncritical activities. First, activities are placed in decreasing resource rate order. Second, if a tie on resource rate exists, the tied activities are placed in increasing order of total float. Third, if there is a tie on both the resource rate and the total float, the priority is to place the activities in decreasing order of sequence steps. The method accepts fractional values for activity durations and for relationships lead times between activities. This heuristic method assumes that the duration of each activity is constant, the resource rate of utilization of the resource is also constant, the activities are time continuous, and the network logic is fixed.

Kobayashi and Nonaka (1990) very briefly described a Hopfield-model-based scheduling method, which minimizes daily resource usage variation by neural computation. The authors compared two approaches to schedule integration of hierarchical scheduling, applicable to large scale plant projects. The first approach is a knowledge-based interactive scheduling method, and the second is based on using ANNs. The comparison between these two approaches suggests that the less automated and demanding knowledge-based method gives a better solution. Although the problem of integrating sub-schedules to plan an overall schedule with flattened resource usage is not identical with the classical resource leveling problem, the concept of using Hopfield NN to flatten the resource usage deserves a special consideration. However, the authors did not provide any implementation details of their work to support their results.

The first use of ANNs in solving the RL problem was attempted by Shimazaki, Sano and Tuchiya (1991). They have formulated the RL as a constrained-optimization problem, and have studied the possibility of mapping this optimization onto a Hopfield NN architecture. However, they did not address the problem of developing an explicit NN model associated with an energy function, that would have required a solution to the key question of specifying the weight-matrix. The mapping is ill-defined, due to the nonlinear form of the constraints, which introduces cross-products of four variables in the expression of their energy function. Since the energy function of a Hopfield NN contains only products of two variables, in order to use the energy function proposed by the authors, an extended set of variables had to be considered instead of the one proposed by the authors. Moreover, the optimal solution presented by the authors does not represent a schedule for the example project, and the objective function finally converged to its initial value. While this approach is interesting from a conceptual point of view, there remains skepticism that it can be used without further and extensive developments, which is the concern of this research work.

2.3 Summary

In this chapter, a review of the literature related to resource leveling has been presented. Different resource leveling models have been discussed, and their advantages as well their limitations have been outlined.

Due to the combinatorial complexity of the RL problem, the utilization of optimization techniques seems to be impractical. The most practical procedures utilize heuristic algorithms. However, these heuristic procedures do not guarantee optimal project schedules, and no heuristic or combinations of heuristic methodologies have demonstrated the ability to consistently produce the best schedule for every project. Because the current techniques (optimization, heuristic) are not sufficient in solving the RL problem, there is a need for other avenues, such as ANNs, to be explored.

Chapter 3

Neural Networks

3.1 Introduction

Neurocomputing is a fundamentally new and different information-processing paradigm and an alternative to algorithmic programming. Neurocomputing can be defined as "the engineering discipline concerned with nonprogrammed adaptive information-processing systems - neural networks - that develop associations between objects in response to their environment" (Hecht-Nielsen 1988). Instead of performing a step-by-step procedure for carrying out the desired transformation, the neural network itself generates its own internal rules governing the association, and refines those rules by comparing its results to the provided examples. Through trial and error, the network learns how to perform the task. Neurocomputing does not, however, replace algorithmic programming. Algorithmic computing and neurocomputing complement each other (Flood and Kartam 1994).

3.2 Neural Networks - General

In contrast to conventional computers which are programmed to perform specific tasks, most neural networks must be taught, or trained. They learn new associations, new patterns, and new functional dependencies. Learning rules and algorithms generated from examples that are used for training the networks, replace the programming required for conventional computation. Unlike programmers using conventional programming methods, NN users do not specify an algorithm to be executed by each computing node. Instead, they select what in their view is the best architecture, specify the characteristics of the neurons and initial weights, and choose the training mode for the network. Appropriate procedures are then input to the network, so it can acquire knowledge from the environment. As a result of such exposure, the network assimilates the information that can later be recalled by the user. Instead of performing a program of instructions sequentially as in a von Neumann computer, NN models explore many competing hypothesis simultaneously using massively parallel networks composed of many computational elements connected by links with variable weights (Lippmann 1987).

Compared to conventional digital computing techniques, neural networks are advantageous because they can learn from examples and generalize solutions, can adapt to fine changes in the nature of a problem, are tolerant to errors in the input data, and can process information rapidly. They are suitable for solving complex pattern- recognition problems, identifying handwritten characters, in adaptive control, for estimation, classification, and generally for problems that have proved intractable or far too expensive with algorithmic computers. However, neural networks have a number of shortcomings such as, the production of inexact solutions, a lack of theory to guide selection of the most appropriate size and configuration for a network, slow progress during training, no guarantee of success in finding an acceptable solution, and a limited ability to rationalize the solutions provided. Although NNs do not require knowledge acquisition, training requires a comprehensive set of

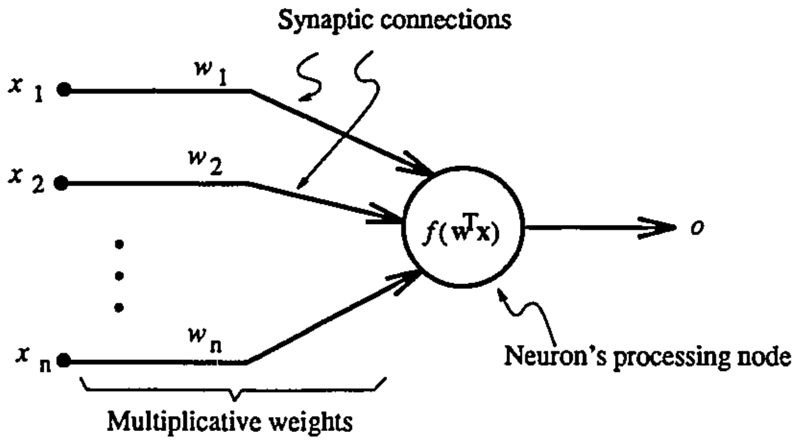


Figure 3.1: General symbol of a neuron (from Zurada 1992, p.32)

examples and the quality of these will affect the NN's final performance. Where appropriate, the advantages of neural networks can be combined with those of symbolic and procedural methods of computation in hybrid schemes (Flood and Kartam 1994).

3.2.1 Processing Elements

The basic processing elements of artificial neural networks are called artificial neurons; they are more often referred to as neurons, nodes, units, or processing elements (PEs). Also, artificial neural networks (ANNs) are usually called neural networks (NNs). All these terms are used interchangeably throughout this thesis. Each processing element (neuron) has many input signals, but only a single output signal. A general neuron symbol is shown in Figure 3.1.

The signal flow of neuron inputs, x_i, is considered to be unidirectional as indicated by arrows, as is the neuron's output signal flow, o. The strength of each connection x_i entering a processing element is indicated by an adaptive coefficient called weight, w_i. This weight is generally used to amplify, attenuate, and possibly change the sign of the signal in the incoming connection.

The processing that each neuron does is determined by a transfer or activation function - a mathematical formula that defines the element's output signal as a function of the input signals and the adaptive coefficients (weights). The activation function is assumed to be nonlinear. Hard limiting (i.e., either the step or the signum function), threshold, and soft limiting (i.e., the sigmoidal) are the three most often used forms of nonlinearities (Vemuri 1988). In some cases, neurons can be considered as simple threshold units that fire when their total input exceeds certain bias levels. The resulting value of the activation function is the output of the artificial neuron, and it is given by the following relationship

$$o = f(\mathbf{w}^{\mathrm{T}}\mathbf{x}) \,, \tag{3.1}$$

where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ is the input vector, and $f(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ is the activation function. Usually, the activation function is referred to as $f(net)$, where the variable net is defined as the scalar product of the weight and input vector

$$net \triangleq \mathbf{w}^{\mathrm{T}}\mathbf{x} \,. \tag{3.2}$$

Neuron inputs and outputs are usually either discrete or continuous, or a mixture of them (Zurada 1992).

3.2.2 Neural Networks Architectures

The output signal of a neuron fans out along many pathways to provide input signals to other neurons. These pathways connect the processing elements into a network, usually called artificial neural network, or simply neural network, as depicted in Figure 3.2.

Often a NN is divided into layers or groups of processing elements, all having the same activation function. The topology of the connections among neurons influences what function a NN can carry out. Over the years, many topological configurations have been proposed. Networks comprising only a single layer of PEs as

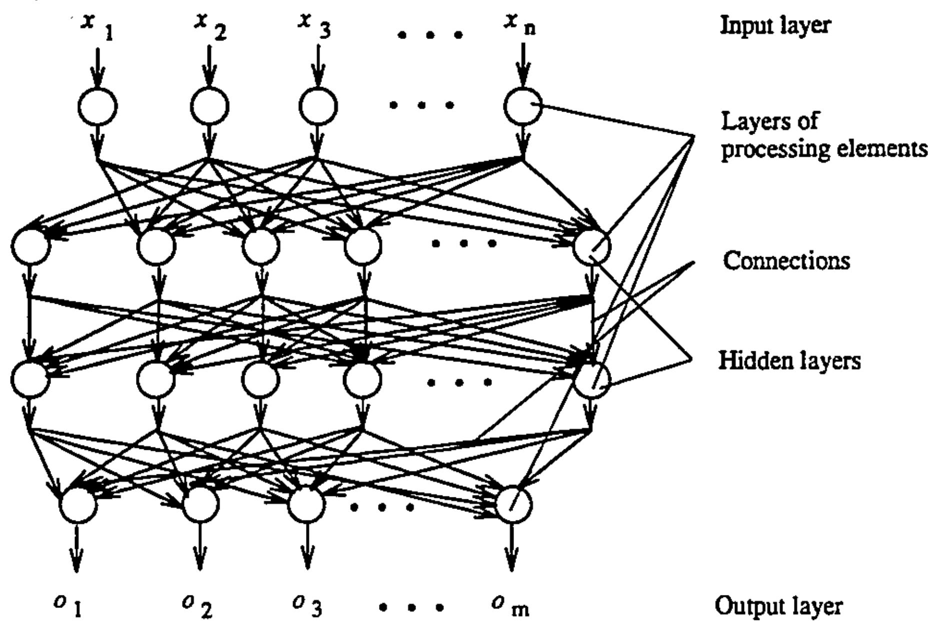


Figure 3.2: General representation of a feedforward neural network (from Hecht-Nielsen 1988, p.37)

well as many layers have been tried. In multi-layer NNs neurons are located in one of three types of places: the input layer, the output layer, and the hidden layer(s). Neurons in the input layer and the output layer are used to communicate with the exterior, while neurons in the hidden layers communicate only with other neurons. In feedforward networks, the signal flow from one layer to the next is unidirectional. In feedback networks, both forward and backward connections exist. Among the various NN paradigms available, the feedforward NN, with backpropagation of the error, is by far the most utilized architecture (Hegazy et al. 1994).

In Figure 3.2, an input array, or sequence of numbers, is entered into the network. Each neuron in the first layer takes a component of the input array, operates on it in parallel with the other neurons in the layer according to the activation function, and delivers a single output to processing elements in the next layer. The result is an output array representing some characteristic associated with the input.

Since inputs and weights can change over time, the network adapts and learns. Most activation functions include a learning law, which is an equation that modifies all or some of the neuron's weights, in response to the input signals and the values supplied by the activation function. The learning law allows the neuron's response to change with time, depending on the nature of the input signals. It is the means by which the network "learns" or adapts itself to the desired outputs. A neural network learns how to process information usually by being given either supervised or unsupervised training. In both cases, it runs through a series of trials. Supervised learning occurs when the NN is supplied by both the input values and the correct output values, and the NN adjusts its weights based upon the error of the computed output. The concept of error in a neural network is defined as the degree to which the observed pattern of output differs from the expected pattern of output. Unsupervised learning occurs when the NN is only provided with the input values, and the network adjusts the weights based solely on the input values and the current network output. An example of unsupervised training applied to civil engineering is that of simulated evolution, which has been used to train a network to provide near-optimal solutions to the flow-shop sequencing problem (Flood 1989).

Neural networks vary in their architecture, each being designed to perform a different type of task. Although all of them consist of processing elements joined by a multitude of connections, they differ in the learning laws incorporated into their activation functions, the topology of their connections, and the weights assigned to their connections. In fact, some neural networks can learn without being trained (self-organizing map) and some do not even learn at all (Hopfield network) (Zurada 1992; Freeman and Skapura 1992). Neural network models have been successfully applied to solve a variety of problems. In particular, Hopfield neural networks have provided acceptable solutions to optimization problems such as A/D conversion, linear programming and the travelling salesperson problem (TSP) (Hopfield and Tank 1985; Tank and Hopfield 1986). In order to explain how NNs can be used in solving optimization problems, the Hopfield NN is presented next.

3.3 Hopfield Neural Network

There are two versions of an ANN, usually called the Hopfield memory, or the Hopfield NN. The two versions are the discrete Hopfield memory, or discrete-time Hopfield network, and the continuous-time Hopfield network, depending on whether the neuron outputs are a discrete or a continuous function of the inputs.

A Hopfield NN is a single layer feedback network with complete interconnections, and it can be viewed as a nonlinear dynamical system. As a dynamical system, it processes the initial condition information over time, while moving through a sequence of states. The general architecture of a Hopfield NN is illustrated in Figure 3.3.

Figure 3.3 shows a fully connected network of n neurons, without the feedback from each neuron to itself. Note also the existence of the external input signals i_i, usually called biases, and the threshold values T_i. The feedback input to the i*th* neuron is equal to the weighted sum of neuron outputs x_j, where $j = 1, 2, \ldots, n$ and $j \neq i$. Denoting w_{ij} as the weight value connecting the output of the j*th* neuron with the input of the i*th* neuron, we can express the total input net_i of the i*th* neuron as

$$net_i = \sum_{j=1, j\neq i}^{n} w_{ij}x_j + i_i - T_i \,, \quad i = 1, 2, \ldots, n \,. \tag{3.3}$$

Equation (3.3) can be rewritten as

$$\mathbf{net} = \mathbf{W}\mathbf{x} + \mathbf{i} - \mathbf{t} \,. \tag{3.4}$$

where

$$\mathbf{x} \triangleq [\, x_1 \, x_2 \, \ldots \, x_j \, \ldots \, x_n \,]^T \,, \tag{3.5}$$

is the output vector,

$$\mathbf{net} \triangleq [\, net_1 \, net_2 \, \ldots \, net_j \, \ldots \, net_n \,]^T \,, \tag{3.6}$$

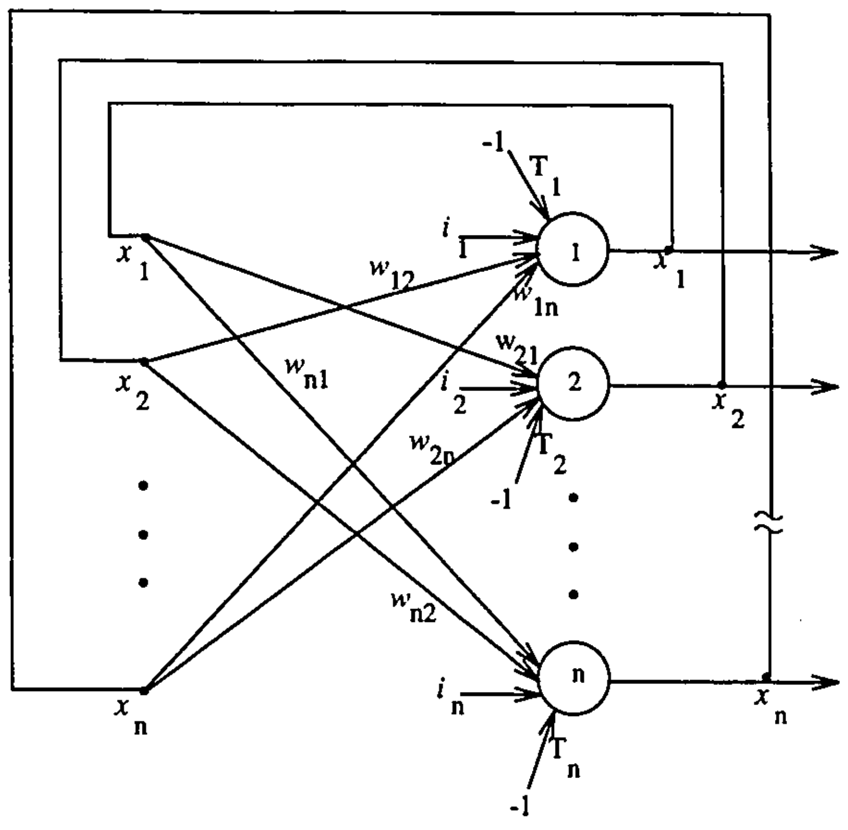


Figure 3.3: Hopfield neural network (from Zurada 1992, p.255)

is the vector containing the activations to each neuron,

$$\mathbf{t} \triangleq [\, \mathrm{T}_1\, \mathrm{T}_2 \, \ldots \, \mathrm{T}_j \, \ldots \, \mathrm{T}_n \,]^{\mathrm{T}} \,, \tag{3.7}$$

is the threshold vector, and

$$\mathbf{i} \triangleq [\, i_1\, i_2 \, \ldots \, i_j \, \ldots \, i_n \,]^{\mathrm{T}} \,, \tag{3.8}$$

is the vector containing the external inputs to each neuron. Matrix **W**, usually called the weight (connectivity matrix), is an $n \times n$ matrix containing the network weights, and can be defined as

$$\mathbf{W} \triangleq \begin{bmatrix} 0 & w_{12} & w_{13} & \ldots & w_{1n} \\ w_{21} & 0 & w_{23} & \ldots & w_{2n} \\ w_{31} & w_{32} & 0 & \ldots & w_{3n} \\ \vdots & & \vdots & \vdots & \vdots \\ w_{n1} & w_{n2} & w_{n3} & \ldots & 0 \end{bmatrix} . \tag{3.9}$$

Note here that, in this model, the weight matrix **W** is symmetrical and with diagonal entries equal to zero.
Typical activation functions used are

$$f(net) \triangleq \frac{1}{1 + exp(-\lambda net)} \,, \tag{3.10}$$

and

$$f(net) \triangleq \mathrm{sgn}(net) = \begin{cases} 1 & , \; net > 0 \\ 0 & , \; net < 0 \,. \end{cases} \tag{3.11}$$

where $\lambda > 0$ in (3.10) is a constant called the gain parameter, and it is proportional to the neuron gain, determining the steepness of the continuous function $f(net)$ near $net = 0$. The continuous activation function is shown in Figure 3.4 for various λ. Notice that as $\lambda \to \infty$, the limit of the continuous function (3.10) becomes the sgn(net) function, defined in (3.11). The soft-limiting activation function (3.10)

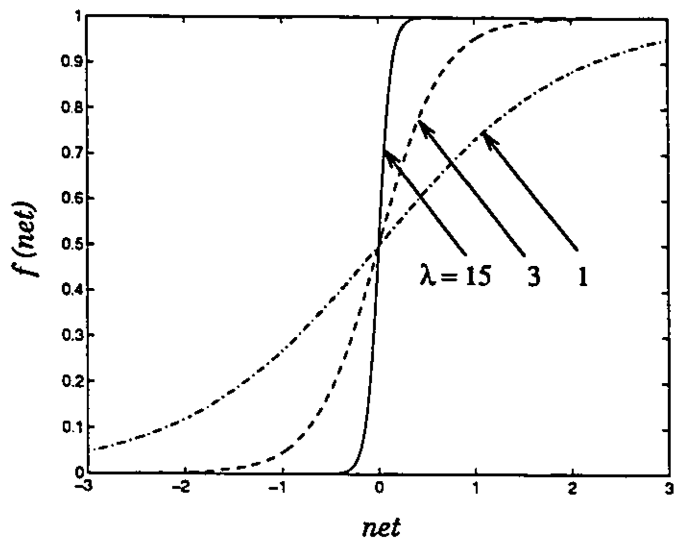


Figure 3.4: Neuron activation function (from Freeman and Skapura 1992, p.145)

is often called sigmoidal characteristic, as opposed to the hard-limiting activation function given in (3.11). The hard-limiting activation function describes the discrete neuron model, while the soft-limiting activation function describes the continuous neuron model.

Corresponding to the unipolar continuous activation function (3.10), the domain of output vector **x** is defined as the interior of the n-dimensional hypercube $(0,1)^n$. When the hard-limiting activation function (3.11) is used, the output vector is one of the vertices of the n-dimensional hypercube $(0,1)^n$. If the asynchronous update scheme is used,

$$x_i = \begin{cases} 1 & , \text{if} \quad net_i > 0 \\ 0 & , \text{if} \quad net_i < 0 \, . \end{cases} \tag{3.12}$$

the output vector moves from vertex to vertex, until it stabilizes in one of the 2^n vertices available. Note that the movement is from a vertex to an adjacent vertex since the asynchronous update mode allows for a single-component update of an

n-tuple vector at a time. The final position of the output vector is determined by weights, thresholds, inputs, the initial vector $\mathbf{x}$, and by the order of transitions. The open question is whether the system is stable, and whether it converges or not.

The stability and the convergence of the Hopfield NN can be evaluated by studying a so-called computational energy function. If this energy function has the properties of a Lyapunov function, i.e., it is positive definite, bounded, and it decreases in time, then the Hopfield NN would be asymptotically stable (Zurada 1992). The energy function for the discrete-time Hopfield NN has the form (Hopfield and Tank 1985)

$$\mathrm{E} = -\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{W}\mathbf{x} - \mathbf{i}^{\mathrm{T}}\mathbf{x} - +\mathbf{t}^{\mathrm{T}}\mathbf{x}\,. \tag{3.13}$$

Hopfield and Tank (1985) have shown that this energy function decreases only when the weight matrix $\mathbf{W}$ is symmetric, i.e., $w_{\mathrm{ij}} = w_{\mathrm{ji}}$. Since the weight matrix $\mathbf{W}$ is indefinite because of its zero diagonal, i.e., $w_{\mathrm{ii}} = 0$, then the energy function E has neither a minimum nor a maximum value in the unconstrained output space. However, since the energy function is bounded within the n-dimensional hypercube, it has to finally reach its minimum in one of the 2^n vertices of the hypercube, under the update algorithm (3.12).

Continuous-time Hopfield NNs are generalized discrete-time Hopfield NNs in which the energy function decreases continuously in time. For a very high gain λ of the neurons, continuous-time networks perform similarly to discrete-time networks, and in the limit case $\lambda \to \infty$, they perform identically. Continuous-time Hopfield NNs are examples of nonlinear, dynamical, and asymptotically stable systems.The evolution of the system is in the general direction of the negative gradient of an energy function. Typically, the network energy function is made equivalent to a certain objective function that needs to be minimized. The search for an energy minimum performed by the NN network corresponds to the search for a solution of an optimization problem. Finding an energy function relevant to a particular optimization problem is a difficult task, and no unique and best method exists to identify such a function.

The suitable energy function for a continuous-time Hopfield NN has the following form (Zurada 1992)

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} - \mathbf{i}^T\mathbf{x} + \sum_{i=1}^{n} G_i \int_0^{x_i} f_i^{-1}(z)dz\,. \qquad (3.14)$$

where$f_i^{-1}(z)$ is the inverse of the activation function f_i. Note that the threshold term of (3.13) has for simplicity been absorbed into the $\mathbf{i}^T\mathbf{x}$ term in (3.14), and the third term containing the integral of the inverse of the activation function has been introduced to account for the property of continuous activation function of neurons. Hopfield and Tank (1985) demonstrated that the evolution of this dynamical system is described by the following set of ordinary nonlinear differential equations with constant coefficients

$$\mathbf{C}\frac{d\mathbf{u}(t)}{dt} = \mathbf{W}\mathbf{v}(t) - \mathbf{G}\mathbf{u}(t) + \mathbf{i} \qquad (3.15)$$

$$\mathbf{v}(t) = \mathbf{f}[\mathbf{u}(t)] \qquad (3.16)$$

where $\mathbf{u}(t)$ denotes a neuron's activation, and $\mathbf{C}$ and $\mathbf{G}$ are notations relevant to the physical model, using electrical components, of the above set of equations, model described in detail in Zurada (1992).

The stable states of the network are completely determined by specifying the bias currents and the initial values of the input voltages, because one could view the sets of differential equations as defining an initial value problem. The matrix $\mathbf{W}$ defines the set of all local minima of the energy function, and the specific values of the steady-state output voltages, obtained from the time evolution of the network, are determined by the initial values of the input voltages, $\mathbf{u}_i$. The equations of motions are deterministic i.e., the steady-state solutions are completely defined by the initial conditions.

Finding an equilibrium point for the dynamic network with continuous activation function neurons corresponds to finding a minimum of $E(\mathbf{x})$. By having previously chosen $w_{ii} = 0$, the weight matrix $\mathbf{W}$ has been made neither positive definite nor negative definite (Zurada 1992), which results from the assumption that the system has no self-feedback. In fact, symmetric matrix $\mathbf{W}$ with zero diagonal produces scalar-valued energy function of quadratic form $(-1/2)\mathbf{x}^T\mathbf{W}\mathbf{x}-\mathbf{i}^T\mathbf{x}$, which has neither minima nor maxima since $\mathbf{W}$ is neither positive nor negative definite. Thus, the energy function $E(\mathbf{x})$ posesses no unconstrained minima. This, in turn, means that in the case of limitation of the output space of the system to a $(0,1)$ hypercube, the constrained minima of $E(\mathbf{x})$ must be located somewhere at the hypercube's boundary, on the edges or faces of the cube. If a solution of the constrained system within the $(0,1)$ cube exists, then it must be a saddle point, which does not represent an attractor for the discussed dynamical system.

Zurada (1992) shows that the assumption of high-gain neurons results in neglecting the last term of energy in (3.14), and enforces the constrained minima to be exactly in the vertices of the $(0,1)$ cube having a total of 2^n vertices. The high-gain network that approaches, in the limit case, the discrete-time system is expected to have its energy minima in the cube corners. For the continuous-time Hopfield NN with finite neuron gain values the minima are within the cube and are attractors of the system. Such minima are usually desirable if they are as close to the vertices of the hypercube as possible. They will always be within the $(0,1)$ cube, and for $\lambda \rightarrow \infty$ they will reach the cube corners. In order to make the weight matrix $\mathbf{W}$ indefinite, and to force the output variables towards the corners of the hypercube, Tank and Hopfield (1986) suggested the adding of additional terms to the energy function, in which the design variables x_i are substituted by $\tilde{x}_i$ given by

$$\tilde{x}_j = x_j(1 - x_j)\,. \tag{3.17}$$

This is especially useful when there is a need for digital solutions (0 or 1) for particular optimization problems.

In conclusion, the equilibrium points of the energy function for a Hopfield NN always represent either the constrained minimum or saddle of the energy function. There is, in general, more than one solution, and the actual solution reached by the NN is dependent on network parameters and on the initial condition within the network. Some of the solutions produced by either discrete or continuous-time Hopfield NNs are useful, so they will be desirable. Other solutions will be less useful and may even be erroneous, and sometimes hard to avoid (Wilson and Pawley 1988).

3.4 Neural Networks in Construction

Neural networks have been considered as promising management supporting tools, since they have capabilities particularly suited for analogy-based decision problems which are relevant at all levels of construction engineering and management (Moselhi et al. 1991). NNs applications in construction began at the end of 1980s (Flood 1989), but already cover a wide range of topics such as, simulation of construction activity (Flood 1989), scheduling (Kobayashi and Nonaka 1990), optimum markup estimation (Moselhi et al. 1991), estimating earthmoving equipment production (Karshenas and Xin 1992), estimating the productivity of various construction activities (Chao and Skibniewski 1994), resource leveling (Shimazaki et al. 1991; Savin et al. 1995), generating project planning networks for construction projects (Mawdesley and Carr 1993), and multiobjective and multiresource decision support systems (Wei and Singh 1995). An extensive review of NNs theory and their applications in civil engineering was carried out by Flood and Kartam (1994), who stressed the important role of NNs in addressing the tasks of interpretation, classification, modelling, prediction, estimation and optimization. Most of the NNs applications in civil engineering have focused on using the feedforward NNs, while little work has been done using feedback NNs (Flood and Kartam 1994). Garret (1992) suggests that feedforward NNs should be used in civil engineering applications for those classes of problems for which the goal state is well known for a given initial state, especially for problems where the initial state descriptions are noisy. On the other hand, feedback NNs (e.g., Hopfield NN) should be used for problems for which the solution is not known in advance. Resource leveling can be included in this class of problems.

3.5 Summary

This chapter presented an introduction to neural networks, in general. Their current applications to civil and construction engineering have been briefly discussed. It also contains a detailed presentation of the Hopfield NN. This NN is a single-layer feedback NN, and it is a dynamical system evolving in time in either a continuous, or discrete, output space. The transition (evolution) in this dynamical NN is toward an asymptotically stable solution, that is a minimum (local or global) of a dissipated energy function. The concept of energy function has been stressed to demonstrate the inherent stability of the Hopfield NNs, and their suitability for producing solutions of certain optimization problems. The stationary solution does not represent, in general, a global optimum solution to the problem of energy function minimization. This is due to the often highly complex shape of the multi-dimensional energy function. This limitation is somewhat moderated by the fact that global solutions of real large-scale minimization problems of energy functions are often mathematically very hard to track anyway, and for large-scale problems cannot be found with certainty.

One of the difficult tasks facing the designers is the translation of the optimization problem into the minimization of an energy function. Moreover, energy functions specific to optimization problems are usually hard to find, and the derivation of the weight matrix $\mathbf{W}$, and of the vector $\mathbf{i}$ of external inputs, is not a straightforward task. The only energy form is the general one stated as in (3.13) or (3.14). If high-gain neurons are used, the third term of the energy expression can be ignored.

Finally, the steps that have to be performed, to map an optimization problem onto a Hopfield NN, can be summarized as follows:

- Choose a representation scheme which allows the outputs of the neurons to be decoded into a solution to the problem
- Find an energy function whose minimum value corresponds to "best" solutions to the problem to be mapped
- Derive the weight matrix and the external inputs vector from the energy function
- Setup the network initial values, which completely determine the stable output of the neurons

There is no direct method for mapping constrained optimization problems onto a NN. However, it can be done through addition of terms in the energy function to penalize the violation of the constraints. The Augmented Lagrangian Multiplier (ALM) method, described in the next chapter, is one of the methods that allows for such a transformation .

Chapter 4

Combinatorial Optimization Using Neural Networks

4.1 Introduction

Artificial neural networks have been recently found to be effective tools to solve different classes of optimization problems (Cichocki and Unbehauen 1993). The ability of analog neuron-like networks to process simultaneously a large number of variables makes it capable of finding solutions for complex optimization problems in a very short period of time. In fact, with the advent of analog VLSI technologies and electrooptics, it is feasible today to design programmable chips which can solve a specific optimization problem considerably faster than by using a sequential algorithm on a general purpose digital computer (Cichocki and Unbehauen 1993).

Many combinatorial optimization problems can be mapped onto neural networks by constructing a suitable energy function. This energy function is later minimized in order to solve the associated systems of differential or difference equations. An approach commonly used for constructing an energy function suitable for mapping a zero-one programming problem onto an ANN (Cichocki and Unbehauen 1993) is briefly reviewed in this chapter.

4.2 Unconstrained Quadratic Zero-One Programming Problems

Many combinatorial optimization problems can be formulated as quadratic 0-1 programming problems. In its simplest form the quadratic 0-1 programming problem can be formulated as follows:

$$\text{minimize}$$

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}\,, \tag{4.1}$$

$$\mathbf{x} \in \{0,1\}^n\,,$$

where $\mathbf{c} \in \Re^n$, $\mathbf{Q} \in \Re^{n\times n}$ is a symmetric $n \times n$ matrix. The components of the design vector $\mathbf{x}$ can take only discrete (binary) values 0 or 1. In other words, the design vector $\mathbf{x}$ is represented by an n-dimensional cube called a unit hypercube with 2^n vertices at the points

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T\,,\ \ x_i \in \{0,1\}\ \ (i = 1,2,\ldots,n)\,.$$

It is to be noted that problem formulation of (4.1) can be written equivalently as

$$\min f(\mathbf{x}) = \mathbf{x}^T\hat{\mathbf{Q}}\mathbf{x}\,,\ \ \mathbf{x} \in \{0,1\}^n\,, \tag{4.2}$$

where $\hat{q}_{ii} = c_i + q_{ii}/2$ and $\hat{q}_{ij} = q_{ij}/2$ for $i \neq j$, since $x_i x_i = x_i{}^2 = x_i$.

In order to solve a quadratic zero-one programming problem using an analog neural network, the discrete variables $x_i \in \{0,1\}$ are replaced by associated continuous variables v_i's ($0 \leq v_i \leq 1$). This means that during the optimization process, any neuron used in the system can change continuously in time its output signal v_i between 0 and 1. However, as $t \to \infty$, all the neurons must be forced to achieve the values of zero or one. In other words, the quadratic zero-one (integer) programming will be transformed into an equivalent minimization problem of a continuous function bounded on the unit hypercube. In the simplest case, when $\hat{\mathbf{Q}}$ is a symmetric negative semidefinite matrix, the zero-one quadratic problem is equivalent to the global optimization problem

$$\text{minimize}$$

$$E_c(\mathbf{v}) = \mathbf{v}^{\mathrm{T}}\hat{\mathbf{Q}}\mathbf{v}\,, \tag{4.3}$$

$$\text{subject to}$$

$$0 \le \mathrm{v}_i \le 1 \;\; (i = 1, 2, \ldots, n)\,.$$

The equivalence of the two problems is based on the fact that the energy (objective) function $E_c(\mathbf{v})$ is concave ($\hat{\mathbf{Q}}$ is negative semidefinite) and therefore it attains its global minimum at a vertex of the hypercube, i.e., $\mathrm{v}_i^* \in \{0, 1\}$. In the general case, when the matrix $\hat{\mathbf{Q}}$ is positive definite or indefinite, an extra penalty term must be added to the energy function to make the energy function concave, i.e., to force the vector $\mathbf{v} = [\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_n]^{\mathrm{T}}$ to converge to a valid solution on the unit hypercube corner. In general, the quadratic zero-one optimization problem (4.1) can be transformed into the equivalent global optimization problem

$$\text{minimize}$$

$$\begin{aligned} E(\mathbf{v}) &= \mathbf{c}^{\mathrm{T}}\mathbf{v} + \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{Q}\mathbf{v} + \frac{\kappa}{2}\mathbf{v}^{\mathrm{T}}(\mathbf{e} - \mathbf{v})\,, \\ &= \sum_{i=1}^{n} \mathrm{c}_i \mathrm{v}_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{q}_{ij}\mathrm{v}_i\mathrm{v}_j + \frac{\kappa}{2}\sum_{i=1}^{n} \mathrm{v}_i(1 - \mathrm{v}_i) \end{aligned} \tag{4.4}$$

$$\text{subject to}$$

$$0 \le \mathrm{v}_i \le 1 \;\; (i = 1, 2, \ldots, n)\,.$$

In (4.4), $\kappa > 0$ is a penalty parameter, and $\mathbf{e} = [1, 1, \ldots, 1]^{\mathrm{T}}$. The last term of the energy function (4.4) is added in order to force the trajectories toward the corners of the unit hypercube. The minimization of the energy function $E(\mathbf{v})$ can be naturally achieved by employing a Hopfield neural network, whose energy function can be written as

$$E(\mathbf{v}) = -\frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{W}\mathbf{v} - \mathbf{t}^{\mathrm{T}}\mathbf{v}\,, \tag{4.5}$$

where $\mathbf{W} = -\mathbf{Q} + \kappa\mathbf{I}$, and $\mathbf{t} = -[\mathbf{c} + (\kappa/2)\mathbf{e}]$.

It is to be noted that the quadratic function (4.4), bounded on the unit hypercube, may have many local minima. In fact, it is possible that all the hypercube corners in the valid subspace are local minima if the function $E(\mathbf{v})$ is concave. The number of local minima strongly depends on the value of the penalty parameter κ. For example, the function $E(\mathbf{v})$ will be concave if we take $\kappa \geq \lambda_{\max} > 0$, where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\mathbf{Q}$. A large number of local minima may severely degrade the performance of the neural network in solving the quadratic optimization problem, since it is very likely that the network will be trapped in a bad local minimum. One of the methods which reduces the overall likelihood that a network will fall into a bad local minimum is to tune the penalty parameter κ during the optimization process in such a way that the energy function $E(\mathbf{v})$ is initially kept as convex as possible, i.e., it contains a possibly small number of local minima. This method is further discussed in what follows, for the specific case of using a Hopfield neural network whose energy function is given by (4.5).

The symmetric $n \times n$ matrix $\mathbf{W}$ of (4.5) can be adjusted as a positive, indefinite or negative definite matrix depending on the value of the penalty parameter κ. Taking the parameter κ negative and its magnitude sufficiently large, the matrix $\mathbf{W}$ will be negative definite and the energy function $E(\mathbf{v})$ will then be convex, i.e., the optimization problem has a single global minimum. If the matrix $\mathbf{W}$ is indefinite, then there will be some local minima. The number of minima will increase with increasing the parameter κ until the matrix $\mathbf{W}$ is positive definite, in which case the energy function $E(\mathbf{v})$ is concave and all the hypercube vertices in the valid subspace are local minima. From the above discussion, it follows that in order to avoid some local minima, the parameter κ should not be kept constant during the optimization process, but rather, it should be gradually increased starting from some negative value at the time $t = 0$, and ending with a positive value $\kappa \geq \lambda_{\max} > 0$, where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\mathbf{Q}$.

4.3 Quadratic 0-1 Optimization with Constraints

In its most general form, a constrained 0-1 optimization problem can be stated as

minimize

$$f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}\,, \tag{4.6}$$

subject to

$$h_i(\mathbf{x}) = 0\,,\ \ i = 1, 2, \ldots, m\,, \tag{4.7}$$

$$g_i(\mathbf{x}) \leq 0\,,\ \ i = m, m+1, \ldots, M\,, \tag{4.8}$$

$$\mathbf{x} \in \{0, 1\}^n\,, \tag{4.9}$$

where $\mathbf{x}$ is the design vector, $f(\mathbf{x})$ is designated as the objective (cost) function, Eqs. (4.7) represent linear equality constraints, and Eqs. (4.8) are linear inequality constraints. In order to make the exposition clear and concise, only the problem of formulating an energy function in the case of an equality-constrained optimization problem will be discussed in the subsequent (i.e., it is assumed that there are no constraints of the form (4.8)). It is also convenient to introduce here a matrix notation for the linear equality constraints of (4.7), i.e.,

$$\mathbf{A}\mathbf{x} = \mathbf{b}\,. \tag{4.10}$$

The energy function corresponding to the optimization problem (4.6) and (4.7) can be conveniently defined as the associated augmented Lagrangian (AL). Specifically, in order to solve an equality-constrained 0-1 programming problem using an ANN, the discrete variables $x_i \in \{0, 1\}$ are replaced by associated continuous variables v_i's ($0 \leq \mathrm{v}_i \leq 1$), and the AL is constructed as

$$E_{\mathrm{L}}(\mathbf{v}) = f(\mathbf{v}) + \sum_{i=1}^{m}\left(\lambda_i h_i(\mathbf{v}) + \frac{1}{2}\kappa_i h_i^2(\mathbf{v})\right)\,. \tag{4.11}$$

Now, following a similar procedure as that one described in Section 4.2, the equality-constrained 0-1 optimization problem (4.6), (4.7) can be transformed into the equivalent global optimization problem

$$E(\mathbf{v}) = f(\mathbf{v}) + \sum_{i=1}^{m} \left(\lambda_i h_i(\mathbf{v}) + \frac{1}{2}\kappa_i h_i^2(\mathbf{v}) \right) + \frac{\kappa}{2} \sum_{i=1}^{n} \mathrm{v}_i(1 - \mathrm{v}_i) \tag{4.12}$$

subject to

$$0 \leq \mathrm{v}_i \leq 1 \;\; (i = 1, 2, \ldots, n)\,, \tag{4.13}$$

where $\kappa > 0$ is a penalty parameter, and $\mathbf{e} = [1, 1, \ldots, 1]^{\mathrm{T}}$. The last term of the energy function (4.12) is added in order to force the trajectories toward the corners of the unit hypercube. The minimization of the energy function $E(\mathbf{v})$ can be naturally achieved by employing a suitable Hopfield neural network. It is to be noted here that, as the design variables are mapped onto neuron outputs, which can be always chosen to assume values between 0 and 1 only, the boundary constraints of (4.13) are actually imposed by the physical realization.

The above discussion suggests that, under ideal circumstances, a solution $\mathbf{v}^*$ to the optimization problem (4.6) and (4.7), could be calculated by a single unconstrained minimization of the differentiable function E of (4.12). Unfortunately, this ideal case can almost never be achieved in practice, since in general, λ_i^* $(i = 1, 2, \ldots, n)$ will not be available until the solution has been found. Hence, an augmented Lagrangian method (ALM) for solving the optimization (4.6) and (4.7), must include a procedure for estimating the Lagrange multipliers. An extensive discussion of the problem of estimating the Lagrange multipliers associated with ALM optimization is provided in (Gill, Murray and Wright 1993). For the purpose of this work, a modified *variable-reduction technique*, to estimate the initial values of the Lagrange multipliers, is used. In this technique, first the constraint matrix $\mathbf{A}$ of (4.10) is partitioned as:

$$\mathbf{A} = [\; \mathbf{V} \;\; \mathbf{U} \;]\,, \tag{4.14}$$

where $\mathbf{V}$ is a non-singular $m \times m$ matrix, and m is the total number of constraints. As a general rule for the RL problem, matrix $\mathbf{V}$ is constructed by taking into account at least one output variable for each noncritical activity. Then, the matrices corresponding to the cost function (without taking into consideration the penalty terms) are similarly partitioned as:

$$\mathbf{Q}_{\mathrm{0f}} = [\ \mathbf{Q}_{\mathrm{0f}}^{*}\ \ \mathbf{Q}_{\mathrm{0f}}^{\sim}\ ]\,, \tag{4.15}$$

where $\mathbf{Q}_{\mathrm{0f}}^{*}$ is a non-singular $m \times m$ matrix, and

$$\mathbf{c}_{\mathrm{0f}} = [\ \mathbf{c}_{\mathrm{0f}}^{*}\ \ \mathbf{c}_{\mathrm{0f}}^{\sim}\ ]\,, \tag{4.16}$$

where $\mathbf{c}_{\mathrm{0f}}^{*}$ is a vector with only m components, corresponding to the selected output variables. First-order multiplier estimates (Gill, Murray and Wright 1993) may be used as an initial estimate of the Lagrange multipliers. Since we wish to find the minimum of the quadratic function

$$\frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{Q}_{\mathrm{0f}}^{*}\mathbf{v} + \mathbf{c}_{\mathrm{0f}}^{*\mathrm{T}}\mathbf{v} + \sum_{i=1}^{m}\lambda_i g_i(\mathbf{v})\,, \tag{4.17}$$

the initial estimates of the Lagrange multipliers can be calculated as the solution to the following linear system:

$$\sum_{i=1}^{m}\frac{\partial g_i(\mathbf{v})}{\partial v_i}\lambda_i^{(0)} = -\ (\mathbf{Q}_{\mathrm{0f}}^{*}\mathbf{v} + \mathbf{c}_{\mathrm{0f}}^{*})\ , \tag{4.18}$$

It is to be noted here that the first-order derivatives of the constraints constitute matrix $\mathbf{V}$ of (4.14). Therefore, the Lagrange multipliers can be calculated as follows:

$$\sum_{i=1}^{m}\mathrm{V}(i,j)\lambda_i^{(0)} = -\ (\mathbf{Q}_{\mathrm{0f}}^{*}\mathbf{v} + \mathbf{c}_{\mathrm{0f}}^{*})\ ,\ j = 1,2,\ldots,m\,. \tag{4.19}$$

During the optimization process, the multipliers are adjusted using a first-order-difference approximation, i.e., at the iteration k,

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + \rho_i \cdot h_i(\mathbf{x}^{(k)})\,. \tag{4.20}$$

In (4.20), ρ_i's are positive scalar variables, and typically $\rho_i = 1$ for all i's.

4.4 Summary

In this chapter, the possibility of using ANNs to solve combinatorial optimization problems was briefly presented and discussed. It was shown that a combinatorial optimization problem can be mapped onto an ANN by constructing a suitable energy function, whose global minimum is in the same time a solution of the optimization problem. In particular, for quadratic integer optimization problems with equality constraints, the associated augmented Lagrangian can be interpreted as the energy function of a Hopfield NN. However, in order to converge to the optimal solution, the augmented Lagrangian method requires a good estimate of the Lagrange multipliers. For the particular case of the RL problem, some helpful suggestions were presented on how to choose the initial values for the Lagrange multipliers.

Chapter 5

Proposed Neural Network Model

5.1 Introduction

Resource leveling problem (RL) can be formulated as an augmented Lagrangian multiplier (ALM) optimization with quadratic cost function and nonlinear constraints (Shimazaki et al. 1991). An attempt to map this nonlinear ALM optimization onto a Hopfield neural network has been reported by Shimazaki et al. (1991). Specifically, Shimazaki et al. (1991) have interpreted the augmented Lagrangian (AL) associated with the RL problem as the energy function of a Hopfield net. However, as shown in Chapter 3, the energy function associated with a Hopfield neural network is a quadratic function. Consequently, the augmented Lagrangian corresponding to a nonlinear ALM optimization, which is not a quadratic function, cannot be mapped directly onto the energy function of a Hopfield neural network, except for the case of some restrictive neighborhoods of the solution to the optimization problem, where the AL is approximately quadratic.

In this chapter, the possibility of formulating an RL problem as a zero-one quadratic ALM (QALM) optimization is investigated[1], and a neural network model

[1]It is to be noted that the AL associated with a zero-one quadratic ALM optimization is a quadratic function, and as a result, it can be naturally identified with the energy function of a Hopfield neural network

of an architecture for construction resource leveling is developed. The derivation of this model is based on the reformulation of the constraints regarding the continuity and the precedence of activities as linear equations.

5.2 Formulation of the RL problem as a Nonlinear ALM Optimization

Shimazaki et al. (1991) have formulated the RL problem as an equality-constrained optimization, in which a quadratic cost function has to be minimized, subject to three types of constraints. They have considered the following cost function

$$f(\mathrm{v}) = \frac{1}{2}\sum_{k=1}^{K}\left(\sum_{j\in\{J_k\}} r_j v_j^{(k)}\right)^2 , \tag{5.1}$$

where the variable $v_j^{(k)}$ may assume values of 1 or 0, according to whether or not the jth activity is being executed during day k, and r_j represents the resource requirement per day for the jth activity. It is to be noted that K designates the project duration in number of days, and the set $\{J_k\}$ consists of all the activities which can take place during day k. The cost function of (5.1) has been traditionally used in solving RL problems, motivated by its simplicity (Carmichael 1989; Harris 1978), as illustrated in Figure 5.1. Denoting by y_k the resource demand for day k, and designating an average (or target) demand over the project duration as y, i.e.,

$$y = \frac{1}{K}\sum_{k=1}^{K} y_k , \tag{5.2}$$

the variance of the daily resource demands is then

$$\sigma_r^2 = \frac{1}{K}\sum_{k=1}^{K}(y_k - y)^2 = \frac{1}{K}\sum_{k=1}^{K} y_k^2 - y^2 . \tag{5.3}$$

Therefore, minimizing the variance σ_r^2 of (5.3) is equivalent to minimizing $\sum_{k=1}^{K} y_k^2$, as the other term (i.e., y^2) is a constant.

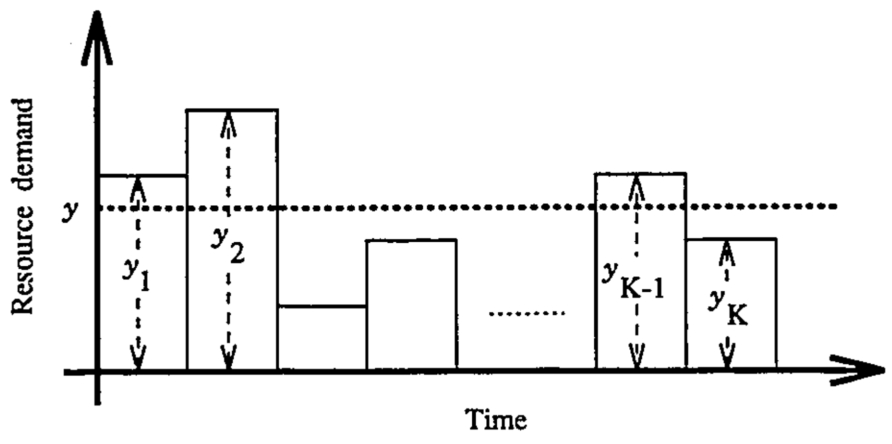


Figure 5.1: Resource plot.

The first type of constraints associated to an RL problem refers to the duration d_j of each activity j, and it can be expressed as

$$\sum_{k=K_1(j)}^{K_2(j)} v_j^{(k)} - d_j = 0 \,, \quad j = 1,2,\ldots,J \,, \tag{5.4}$$

where J denotes the total number of activities, and $K_1(j)$ and $K_2(j)$ are given by

$$K_1(j) \triangleq EST_j + 1 \,, \tag{5.5}$$

$$K_2(j) \triangleq EST_j + d_j + TF_j \,. \tag{5.6}$$

In (5.5) and (5.6), EST_j and TF_j designate, respectively, the earliest start time and the total float of activity j. Let equation (5.4) be denoted by the following notation:

$$\hat{g}_j^{(1)}(v) \triangleq \sum_{k=K_1(j)}^{K_2(j)} v_j^{(k)} - d_j \,, \quad j = 1,2,\ldots,J \,. \tag{5.7}$$

This notation will simplify the presentation of subsequent mathematical expressions.

The main assumptions regarding the activities, their resources and the project logic, are: (1) once an activity is started, it cannot be interrrupted until it is completed; (2) the durations of the activities and the network logic are assumed fixed;

(3) resource requirement of each activity is assumed to be constant for the duration of the activity; and (4) the project completion date is assumed fixed.

The second constraint is related to the continuity of an activity. These continuity constraints assume that an activity cannot be further divided into smaller units, and have been expressed as

$$\sum_{\mathrm{k}=\mathrm{K}_1(\mathrm{j})}^{\mathrm{K}_2(\mathrm{j})-1} v_{\mathrm{j}}^{(\mathrm{k})} v_{\mathrm{j}}^{(\mathrm{k}+1)} = d_{\mathrm{j}} - 1 \,, \quad \mathrm{j} = 1, 2, \ldots, J \,. \tag{5.8}$$

A convenient notation related to (5.8) is introduced here, i.e.,

$$\hat{g}_{\mathrm{j}}^{(2)}(\mathrm{v}) \triangleq \sum_{\mathrm{k}=\mathrm{K}_1(\mathrm{j})}^{\mathrm{K}_2(\mathrm{j})-1} v_{\mathrm{j}}^{(\mathrm{k})} v_{\mathrm{j}}^{(\mathrm{k}+1)} + (1 - d_{\mathrm{j}}) \,, \quad \mathrm{j} = 1, 2, \ldots, J \,. \tag{5.9}$$

The third type of constraints related to an RL problem enforces the precedence relations among the activities within a project-network. Specifically, since for each activity n there is a set of activities $\{p(n)\} = \{m_1, \ldots, m_\ell, \ldots, m_{p(n)}\}$ that must complete before n can start, these precedence constraints can be expressed as

$$\sum_{\mathrm{k}=\mathrm{K}_1(m_\ell)}^{\mathrm{K}_2(m_\ell)} v_{m_\ell}^{(\mathrm{k})} v_n^{(\mathrm{k})} = 0 \quad \text{for all } m_\ell \in \{p(n)\},\ n = 1, 2, \ldots, J \,. \tag{5.10}$$

It is again convenient to introduce a notation related to (5.10) as

$$\hat{g}_{\mathrm{i}}^{(3)}(\mathrm{v}) \triangleq \sum_{\mathrm{k}=\mathrm{K}_1(m_\ell)}^{\mathrm{K}_2(m_\ell)} v_{m_\ell}^{(\mathrm{k})} v_n^{(\mathrm{k})} \,, \quad \mathrm{i} = 1, 2, \ldots, I \,, \tag{5.11}$$

where I designates the total number of precedence constraints in a given project-network, i.e.,

$$I = J(p(1) + p(2) + \ldots + p(J)) \,, \tag{5.12}$$

and the index i can be defined as

$$\mathrm{i} = \begin{cases} \ell & \text{for} \quad n = 1 \\ \ell + \sum_{q=1}^{n-1} p(q) & \text{for} \quad n = 2, 3, \ldots, J \,. \end{cases} \tag{5.13}$$

Having defined the constraints associated with an RL problem as given by (5.4), (5.8) and (5.10), and following a standard procedure of ALM optimization

(Bertsekas, 1982), the equality-constrained nonlinear optimization problem (5.1), (5.4), (5.8) and (5.10) can be solved via the unconstrained minimization of the augmented Lagrangian

$$\begin{aligned} L_S(v) = f(v) &+ \sum_{j=1}^{J} \lambda_j \hat{g}_j^{(1)}(v) + \frac{1}{2}\sum_{j=1}^{J} \alpha_j [\hat{g}_j^{(1)}(v)]^2 + \\ &+ \sum_{j=1}^{J} \mu_j \hat{g}_j^{(2)}(v) + \frac{1}{2}\sum_{j=1}^{J} \beta_j [\hat{g}_j^{(2)}(v)]^2 + \\ &+ \sum_{i=1}^{I} \nu_i \hat{g}_i^{(3)}(v) + \frac{1}{2}\sum_{i=1}^{I} \gamma_i [\hat{g}_i^{(3)}(v)]^2 , \end{aligned} \tag{5.14}$$

where λ_j's, μ_j's and ν_i's denote the Lagrange multipliers, while α_j's, β_j's and γ_i's represent the associated penalty parameters. Note that by constructing the AL as given by (5.14), Shimazaki et al. (1991) did not impose the condition that $v_j^{(k)}$'s be integer (i.e., $v_j^{(k)} \in \{0,1\}$ for all j's and k's). However, the solutions were still expected to be integer as a result of mapping the AL of (5.14) onto the energy function of a Hopfield network, and then employing this network to find the solution to the original RL problem. In this thesis, it is observed that, in fact, the operation of mapping the AL of (5.14) onto the energy function of a Hopfield net is actually not always possible, in view of the nonlinearity of the terms $\sum_{j=1}^{J} \beta_j [\hat{g}_j^{(2)}(v)]^2$ and $\sum_{i=1}^{I} \gamma_i [\hat{g}_i^{(3)}(v)]^2$ of (5.14). As an example, the term $[\hat{g}_j^{(2)}(v)]^2$ contains nonlinear crossproducts of the form

$$v_j^{(k)} v_j^{(k+1)} v_j^{(k')} v_j^{(k'+1)} . \tag{5.15}$$

In order to overcome the above mentioned drawback of the nonlinear ALM optimization, an alternative formulation of the RL problem as a 0-1 QALM optimization is developed in the subsequent.

5.3 Formulation of the RL problem as a QALM Optimization

The major drawback associated with the nonlinear ALM optimization can be eliminated by deriving, when possible, linear expressions for the constraints regarding the continuity and the precedence of the activities of a project-network. This possibility is discussed below.

5.3.1 Derivation of a Linear Expression for the Constraint Regarding the Continuity of an Activity

A procedure for deriving a linear expression for the constraint regarding the continuity of an activity j in the case when the total float TF_j is smaller than or equal to the duration d_j is illustrated in Figure 5.2. Let us assume that the optimal start time for the *jth* activity is $K_f - 1$, as shown in Figure 5.2(b). The position $K_f - 1$ of the *jth* activity on an optimal schedule can be determined as

$$K_f - 1 = \sum_{k=K_1(j)}^{K_3(j)-1} \left[1 - v_j^{(k)}\right] , \tag{5.16}$$

where $K_3(j)$ is a notation for $K_1(j) + d_j$, i.e.,

$$K_3(j) \triangleq K_1(j) + d_j \,. \tag{5.17}$$

Using (5.16), and observing that $\sum_{k=K_1(j)}^{K_3(j)-1} 1 = d_j$, we can express K_f as

$$K_f = 1 + d_j - \sum_{k=K_1(j)}^{K_3(j)-1} v_j^{(k)} \,. \tag{5.18}$$

At this point, it is observed that *jth* activity is continuous if and only if the following relation holds

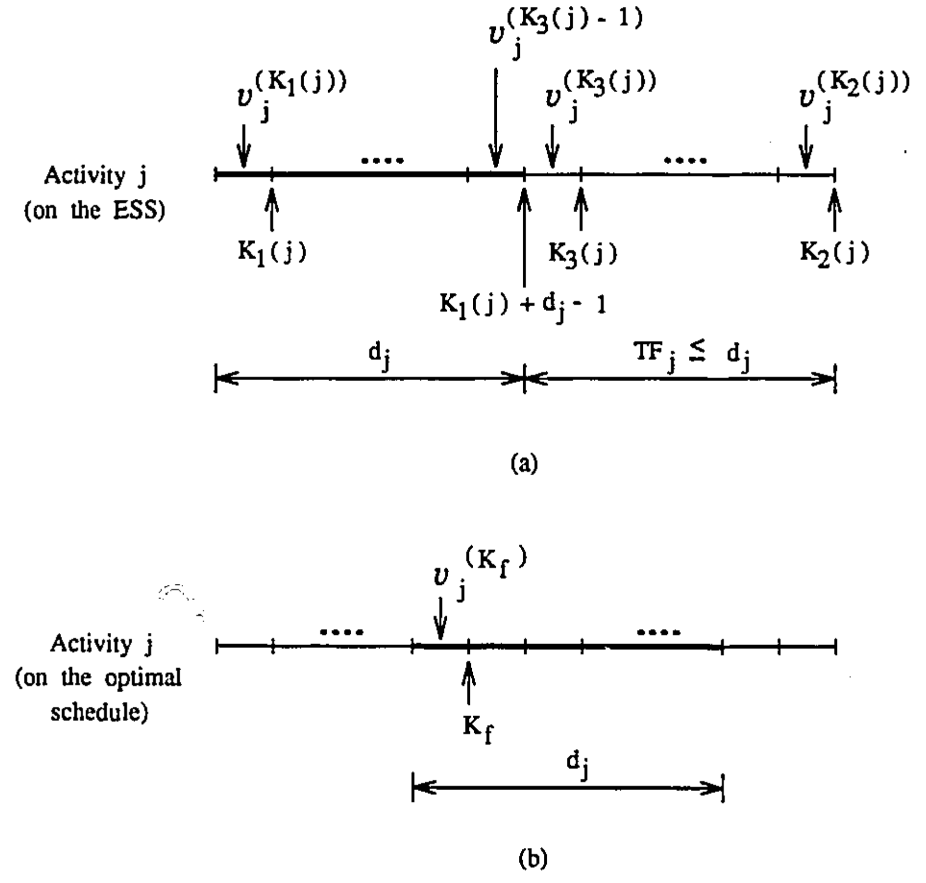


Figure 5.2: An illustration of the procedure for the derivation of a linear expression for the continuity constraint in the case when $\mathrm{TF}_j \leq d_j$. (a) The position of the jth activity on the ESS. (b) The position of the jth activity on an optimal schedule.

$$\sum_{k=K_1(j)}^{K_2(j)} kv_j^{(k)} = K_f v_j^{(K_f)} + (K_f+1)v_j^{(K_f+1)} + \ldots + (K_f + d_j - 1)v_j^{(K_f+d_j-1)}$$

$$= K_f \sum_{k=K_f}^{K_f+d_j-1} v_j^{(k)} + \sum_{k=1}^{d_j-1} kv_j^{(K_f+k)} . \tag{5.19}$$

Observing that $\sum_{k=K_f}^{K_f+d_j-1} v_j^{(k)} = d_j$, while

$$\sum_{k=1}^{d_j-1} kv_j^{(K_f+k)} = 1+2+3+\ldots+(d_j-1) , \tag{5.20}$$

(5.19) becomes

$$\sum_{k=K_1(j)}^{K_2(j)} kv_j^{(k)} = d_j K_f + \frac{d_j(d_j-1)}{2} . \tag{5.21}$$

On substituting K_f as given by (5.18) into (5.21), we get

$$\sum_{k=K_1(j)}^{K_2(j)} kv_j^{(k)} = d_j(d_j+1) - \sum_{k=K_1(j)}^{K_3(j)-1} d_j v_j^{(k)} + \frac{1}{2} d_j(d_j-1) . \tag{5.22}$$

Consequently, the continuity constraint for an activity j for which $TF_j \leq d_j$ can be written as

$$\sum_{k=K_1(j)}^{K_3(j)-1} (k+d_j)v_j^{(k)} + \sum_{k=K_3(j)}^{K_2(j)} kv_j^{(k)} - \delta_j = 0 , \tag{5.23}$$

where $\delta_j \triangleq \frac{1}{2} d_j(3d_j+1)$.

The procedure described above demonstrates that it is indeed possible to replace the nonlinear expression of the constraint imposing the continuity of an activity by a linear equation. However, this applies only to those activities for which $d_j \geq TF_j$. In the general case of a project-network which has activities with $d_j < TF_j$, the following example illustrates a very good strategy for using the same linear equation.

Example 5.1 Let us consider the simple project-network whose diagram is illustrated in Figure 5.3(a). The early start schedule (ESS) and the corresponding histogram of required resources for the project are shown in Figures 5.3(b) and 5.3(c), respectively. As it can be observed from Figure 5.3(b), in the case of activity A (i.e., j = 1), $d_1 = 2$ while $\mathrm{TF}_1 = 4$. While employing equation (5.23) to solve the RL problem for this project-network through optimization, we observe that there are 5 possible outcomes for the starting point of activity A on the optimal schedule, as illustrated in Figure 5.3(d). It can be easily noted that the solution to this RL problem can be obtained by carying out the optimization procedure for two project-schedules of lower complexity PS_1 and PS_2 (see Figures 5.4(a) and 5.4(b)) in parallel, and then selecting the solution that yields the minimum cost. The optimal schedule for the project-network of Figure 5.3(a) and the corresponding histogram of required resources are shown in Figures 5.4(c) and 5.4(d), respectively. In this example, there is no need for the continuity constraint to be imposed for the activities B and C, as their durations are both equal to 1.

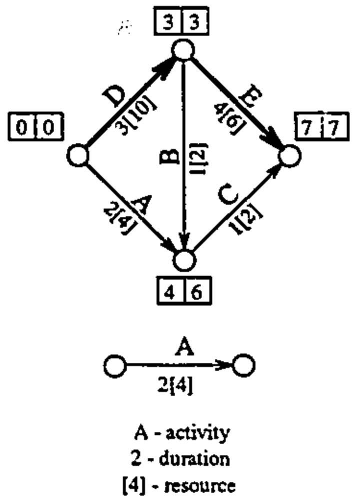
Thus, it is possible to solve the RL problem for an arbitrary project-network, which has activities with $d_j < \mathrm{TF}_j$, by analyzing project-schedules of lower complexity, for which $d_j \geq \mathrm{TF}_j$ for all j's. As a result, only the cases of RL problems for which the continuity constraint can be expressed in a linear form will be investigated in the following sections.

5.3.2 Derivation of a Linear Expression for the Constraint Regarding the Precedence of Activities

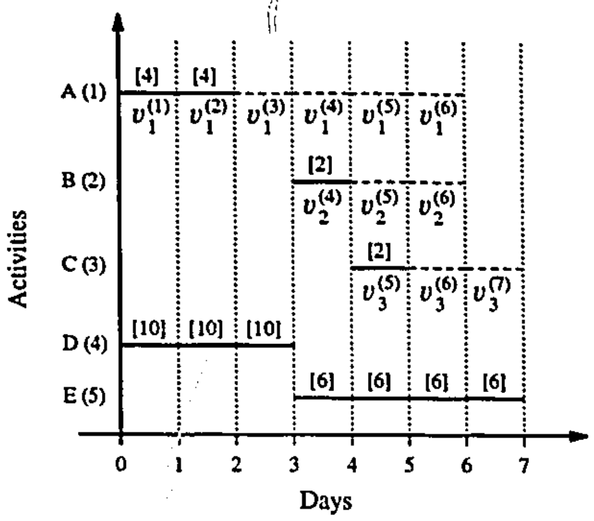
Unlike the case of the continuity constraint, deriving a linear expression related to the precedence constraint is straightforward, as is described in what follows.

Let i denote an activity which has to be completed before the beginning of activity j, as illustrated in Figure 5.5. The activity i precedes activity j if and only if they do not overlap over the interval $[\mathrm{K}_1(\mathrm{j}), \mathrm{K}_2(\mathrm{i})]$. Since the continuity constraint for each of the activities i and j has already been imposed, the precedence constraint can simply be expressed as

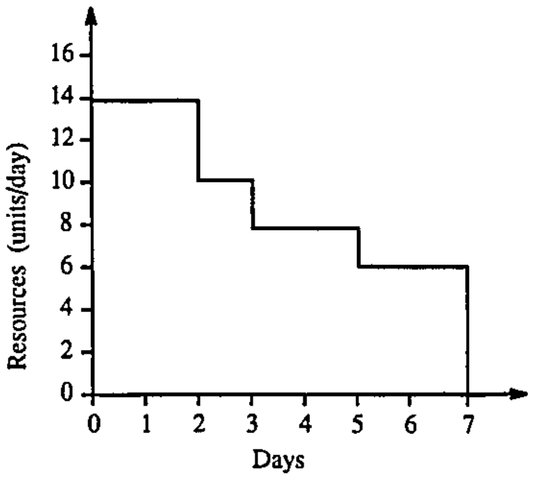
$$v_i^k + v_j^k \leq 1\,, \quad \mathrm{k} = \mathrm{K}_1(\mathrm{j}), \mathrm{K}_1(\mathrm{j})+1, \ldots, \mathrm{K}_2(\mathrm{i})\,. \tag{5.24}$$



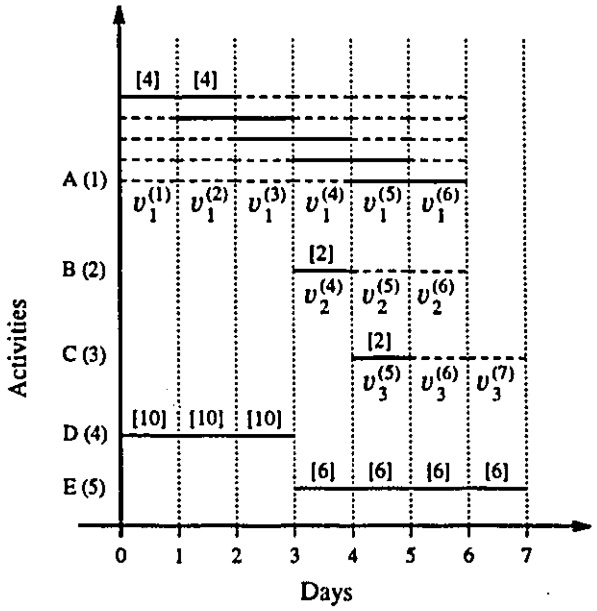
(a) Activity-on-arrow diagram



(b) Early start schedule

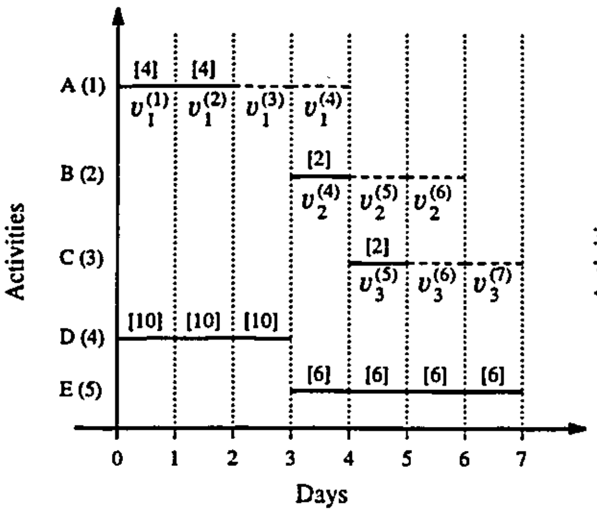


(c) Histogram of required resources before leveling

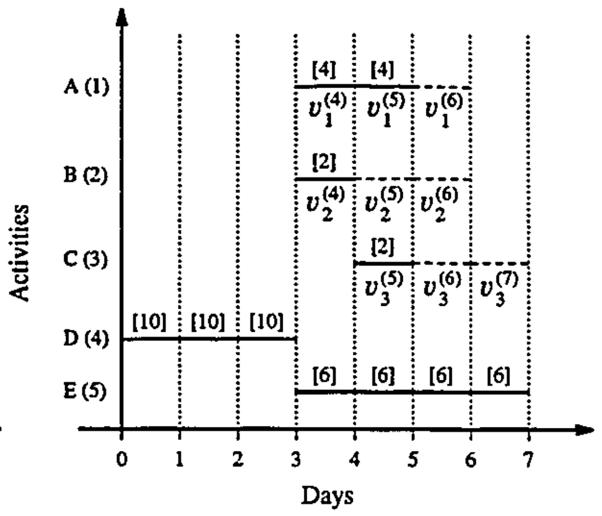


(d) Bar-chart illustrating all the possible solutions for activity A (1)

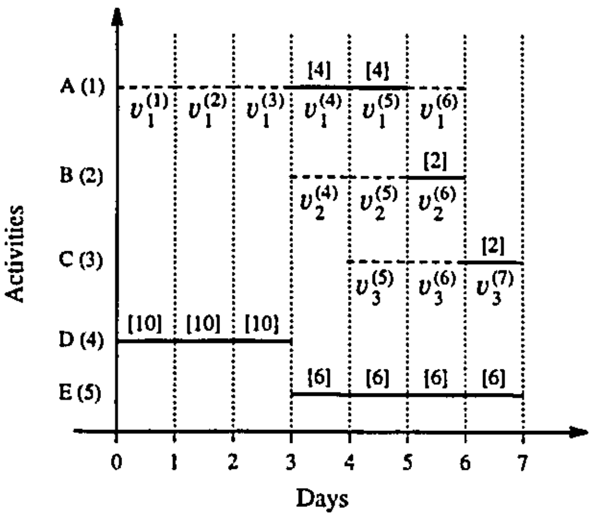
Figure 5.3: An example of a project-network which has activities with $d_j < \mathrm{TF}_j$.


A (1)
B (2)
C (3)
D (4)
E (5)
Activities
[4]
[2]
[10]
[6]
$v_1^{(1)}$ $v_1^{(2)}$ $v_1^{(3)}$ $v_1^{(4)}$
$v_2^{(4)}$ $v_2^{(5)}$ $v_2^{(6)}$
$v_3^{(5)}$ $v_3^{(6)}$ $v_3^{(7)}$
0 1 2 3 4 5 6 7
Days

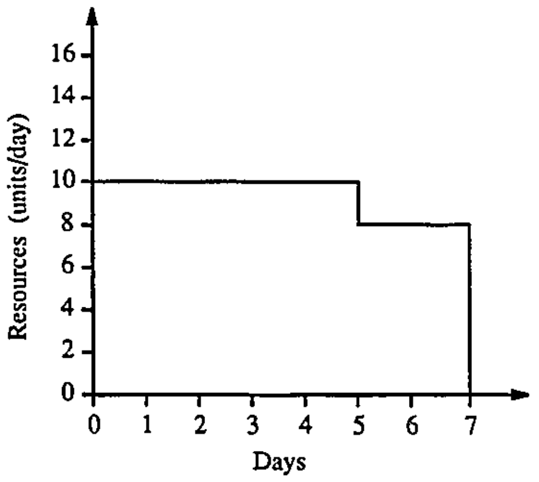
(a) Project-schedule PS_1


A (1)
B (2)
C (3)
D (4)
E (5)
Activities
[4]
[2]
[10]
[6]
$v_1^{(4)}$ $v_1^{(5)}$ $v_1^{(6)}$
$v_2^{(4)}$ $v_2^{(5)}$ $v_2^{(6)}$
$v_3^{(5)}$ $v_3^{(6)}$ $v_3^{(7)}$
0 1 2 3 4 5 6 7
Days

(b) Project-schedule PS_2


A (1)
B (2)
C (3)
D (4)
E (5)
Activities
[4]
[2]
[10]
[6]
$v_1^{(1)}$ $v_1^{(2)}$ $v_1^{(3)}$ $v_1^{(4)}$ $v_1^{(5)}$ $v_1^{(6)}$
$v_2^{(4)}$ $v_2^{(5)}$ $v_2^{(6)}$
$v_3^{(5)}$ $v_3^{(6)}$ $v_3^{(7)}$
0 1 2 3 4 5 6 7
Days

(c) Optimal schedule



(d) Histogram of required resources after leveling

Figure 5.4: An illustration of reducing the solution to the RL problem associated with the project-network of Figure 5.3 by carying out the optimization procedure for two project-schedules of lower complexity: PS_1 and PS_2.

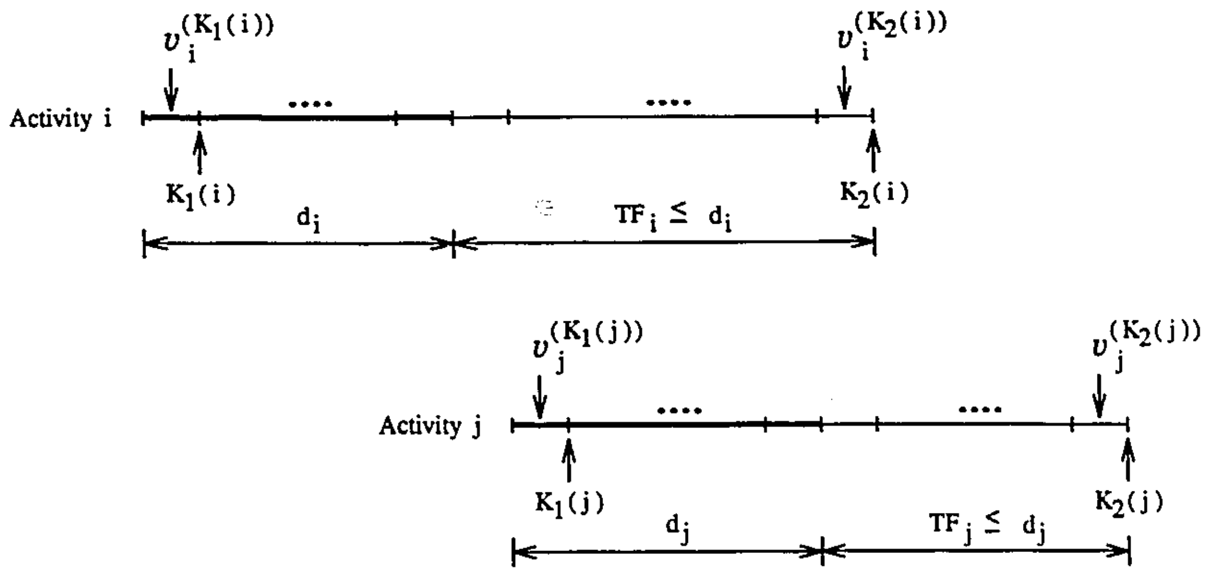


Figure 5.5: An illustration of the relation of precedence between two activities

An alternative expression for imposing the precedence constraint (5.24) could be

$$v_i^{(k)} v_j^{(k)} = 0\,, \quad k = K_1(j), K_1(j)+1, \ldots, K_2(i)\,. \tag{5.25}$$

Although the expression (5.25) is nonlinear, it can still be used for the formulation of a QALM optimization for resource leveling. Specifically, in view of the methodology described in Section 4.3, due to the presence of the penalty term

$$\frac{1}{2}\,\tilde{\alpha}\, v_i^{(k)} v_j^{(k)} \left[\, 1 - v_i^{(k)} v_j^{(k)} \,\right] \tag{5.26}$$

in the expression 4.12 of the augmented Lagrangian, the nonlinear contribution

$$\frac{1}{2}\,\alpha \left[\, v_i^{(k)} v_j^{(k)} \,\right]^2 \tag{5.27}$$

of the term $v_i^{(k)} v_j^{(k)}$ can be cancelled out by making $\alpha = \tilde{\alpha}$.

5.3.3 Proposed Formulation of the RL Problem as a QALM Optimization

Having introduced suitable expressions for the constraints regarding the continuity and precedence of the activities of a project-network, we will now formulate the RL problem as a QALM optimization. Let us first assume that, in the RL problem, the activities which belong to a critical path of a project-network[2] would not introduce additional design variables $v_j^{(k)}$'s. As a result, it is convenient to reformulate the cost function $f(\mathbf{v})$ of (5.1) as

$$f(\mathbf{v}) = \frac{1}{2}\sum_{k=1}^{K}\left(\sum_{j\in\mathcal{J}_k} r_j v_j^{(k)} + \sum_{j\in\mathcal{J}_k^*} r_j\right)^2 . \tag{5.28}$$

In (5.28), a variable $v_j^{(k)}$ may assume values of 1 or 0 according to whether or not the jth activity can be executed during day k, and r_j represents the resource requirement per day for the jth activity. The project duration is denoted by K, while the notation $\mathcal{J}_k$ designates a set consisting of all the noncritical activities which can take place during day k. Similarly, $\mathcal{J}_k^*$ denotes a set which consists of all the critical activities that can take place during day k.

Now, the first type of the constraints associated with the RL problem, which refers to the duration d_j of each activity j, can be expressed as

$$g_j^{(1)}(\mathbf{v}) = 0 , \qquad j = 1, 2, \ldots, J , \tag{5.29}$$

where

$$g_j^{(1)}(\mathbf{v}) \triangleq \sum_{k=K_1(j)}^{K_2(j)} v_j^{(k)} - d_j , \qquad j = 1, 2, \ldots, J , \tag{5.30}$$

with J denoting the total number of noncritical activities, while $K_1(j)$ and $K_2(j)$ are given by (5.5) and (5.6), respectively.

[2]Note that all the activities which belong to a critical-path of a project-network are called critical activities. All other activities of the project-network are designated as noncritical

Similarly, the continuity constraints associated with the RL problem can be expressed as

$$g_j^{(2)}(\mathrm{v}) = 0\,, \qquad j = 1,2,\ldots,\mathrm{J}\,, \tag{5.31}$$

with

$$g_j^{(2)}(\mathrm{v}) \triangleq \sum_{k=K_1(j)}^{K_3(j)-1} (k + d_j) v_j^{(k)} + \sum_{k=K_3(j)}^{K_2(j)} k v_j^{(k)} - \delta_j\,, \tag{5.32}$$

where $\delta_j \triangleq \frac{1}{2} d_j(3d_j + 1)$, and $K_3(j)$ is given by (5.17).

The constraints enforcing the precedence relations among the activities of a project-network are expressed according to the results of Section 5.3.2 (see (5.25)). Specifically, since for each activity j there is a set of activities $\mathcal{P}_j = \{1,2,\ldots,P_j\}$ that must be completed before j can be started, the precedence relations can be imposed as

$$v_p^{(k)} v_j^{(k)} = 0\,, \tag{5.33}$$

for all $k \in \{K_1(j), K_1(j) + 1, \ldots, K_2(p)\}$, $p \in \mathcal{P}_j$, and $j = 1,2,\ldots,J$. As for the previous types of constraints, we introduce the notation

$$g_i^{(3)}(\mathrm{v}) \triangleq v_p^{(k)} v_j^{(k)}\,, \quad i \triangleq i(j,p,k)\,, \quad i = 1,2,\ldots,I\,, \tag{5.34}$$

where I stands for the total number of precedence constraints in a given project-network. Having defined the constraints associated with a RL problem as given by (5.29), (5.31) and (5.33), and following a standard procedure of ALM optimization (Bertsekas 1982, Gill et al. 1993), the equality-constrained quadratic optimization of (5.28), (5.29), (5.31) and (5.33) can be replaced by the unconstrained minimization of the augmented Lagrangian

$$\begin{aligned} L(\mathrm{v}) = f(\mathrm{v}) + \sum_{j=1}^{J} \lambda_j g_j^{(1)}(\mathrm{v}) + \frac{1}{2}\sum_{j=1}^{J} \alpha_j [g_j^{(1)}(\mathrm{v})]^2 + \\ + \sum_{j=1}^{J} \mu_j g_j^{(2)}(\mathrm{v}) + \frac{1}{2}\sum_{j=1}^{J} \beta_j [g_j^{(2)}(\mathrm{v})]^2 + \\ + \sum_{i=1}^{I} \nu_i g_i^{(3)}(\mathrm{v}) + \frac{1}{2}\sum_{i=1}^{I} \gamma_i [g_i^{(3)}(\mathrm{v})]^2\,, \end{aligned} \tag{5.35}$$

where λ_js, μ_js and ν_is denote the Lagrange multipliers, while α_js, β_js and γ_is represent the associated penalty parameters.

By minimizing the augmented Lagrangian (5.35), a solution to the quadratic programming problem (5.28), (5.29), (5.31), (5.33) is obtained. In order to derive a Lagrangian that corresponds to the original RL problem, which is a 0-1 quadratic optimization, it is common practice (Zurada 1992, Cichocki and Unbehauen 1993) to augment the AL of (5.35) by the following additional penalty terms:

$$\tilde{f}_k(\mathbf{v}) \triangleq \frac{1}{2} \sum_{j \in \mathcal{J}_k} r_j^2 \tilde{v}_j^{(k)}, \quad k = 1, 2, \ldots, K, \tag{5.36}$$

$$\tilde{h}_j^{(1)}(\mathbf{v}) \triangleq \frac{1}{2} \alpha_j \sum_{k=K_1(j)}^{K_2(j)} \tilde{v}_j^{(k)}, \quad j = 1, 2, \ldots, J, \tag{5.37}$$

$$\tilde{h}_j^{(2)}(\mathbf{v}) \triangleq \frac{1}{2} \beta_j \left(\sum_{k=K_1(j)}^{K_3(j)-1} (k + d_j)^2 \tilde{v}_j^{(k)} + \sum_{k=K_3(j)}^{K_2(j)} k^2 \tilde{v}_j^{(k)} \right), \tag{5.38}$$

$j = 1, 2, \ldots, J$, and

$$\tilde{h}_i^{(3)}(\mathbf{v}) \triangleq \frac{1}{2} \gamma_i \, v_p^{(k)} v_j^{(k)} (1 - v_p^{(k)} v_j^{(k)}), \tag{5.39}$$

with $i = 1, 2, \ldots, I$. In (5.36)-(5.38), the following notation has been used:

$$\tilde{v}_j^{(k)} = v_j^{(k)} (1 - v_j^{(k)}). \tag{5.40}$$

Consequently, a solution to an RL problem can be obtained by minimizing the AL function

$$E = L + \sum_{k=1}^{K} \tilde{f}_k + \sum_{j=1}^{J} \tilde{h}_j^{(1)} + \sum_{j=1}^{J} \tilde{h}_j^{(2)} + \sum_{i=1}^{I} \tilde{h}_i^{(3)}, \tag{5.41}$$

where L can be calculated using equation (5.35). Note that in the equation (5.41), it is implicitly assumed that all the terms are functions of the design vector $\mathbf{v}$, which is defined as

$$\mathbf{v} = [\, \mathbf{v}_1 \; \mathbf{v}_2 \; \ldots \; \mathbf{v}_j \; \ldots \; \mathbf{v}_J \,]^T, \tag{5.42}$$

with

$$\mathbf{v}_j = \left[\, v_j^{(K_1(j))} \; v_j^{(K_1(j)+1)} \; \ldots \; v_j^{(K_2(j))} \,\right]. \tag{5.43}$$

It is also useful to introduce here some additional notations related to the augmented Lagrangian L of equation (5.35). Specifically, let

$$f_k(\mathrm{v}) \triangleq \frac{1}{2}\left(\sum_{j\in\mathcal{J}_k} r_j v_j^{(k)} + \sum_{j\in\mathcal{J}_k^*} r_j\right)^2 , \quad k = 1,2,\ldots,K , \tag{5.44}$$

$$h_j^{(1)}(\mathrm{v}) \triangleq \frac{1}{2}\alpha_j\,[g_j^{(1)}(\mathrm{v})]^2 , \quad j = 1,2,\ldots,J , \tag{5.45}$$

$$h_j^{(2)}(\mathrm{v}) \triangleq \frac{1}{2}\beta_j\,[g_j^{(2)}(\mathrm{v})]^2 , \quad j = 1,2,\ldots,J , \tag{5.46}$$

$$h_i^{(3)}(\mathrm{v}) \triangleq \frac{1}{2}\gamma_i\,[g_i^{(3)}(\mathrm{v})]^2 , \quad i = 1,2,\ldots,I . \tag{5.47}$$

With these additional notations, equation (5.35) can be expressed as

$$\mathrm{L} = \sum_{k=1}^{K} f_k + \sum_{j=1}^{J} h_j^{(1)} + \sum_{j=1}^{J} h_j^{(2)} + \sum_{i=1}^{I} h_i^{(3)} + \\ + \sum_{j=1}^{J} \lambda_j g_j^{(1)} + \sum_{j=1}^{J} \mu_j g_j^{(2)} + \sum_{i=1}^{I} \nu_i g_i^{(3)} . \tag{5.48}$$

The problem of mapping the QALM optimization of this section onto a Hopfield-network-based architecture is investigated in the following section.

5.4 Mapping the QALM Optimization Onto a Neural Network Architecture

The augmented Lagrangian E of equation (5.41) (with L given by (5.48)) can be written in a matrix form as

$$\mathrm{E}(\mathrm{v}) = \mathbf{c}^T\mathrm{v} + \frac{1}{2}\mathrm{v}^T\mathbf{Q}\mathrm{v} + \mathcal{C} , \tag{5.49}$$

where $\mathcal{C}$ is a constant, and the unconstrained optimization problem that consists of minimizing the augmented Lagrangian E of (5.49) can be solved by employing the proposed ANN architecture shown in Figure 5.6. The proposed ANN configuration consists of two main blocks: the Hopfield NN block, and a control block for the computation of the weights of the Hopfield network, and for updating these weights according to the adjustment of the Lagrange multipliers in the QALM optimization. As discussed in Chapter 3, the dynamics of the Hopfield neural network results in minimizing the energy function $E_H(\mathbf{x})$, which is a quadratic form in the output vector $\mathbf{x}$, i.e.,

$$E_H(\mathbf{x}) = -\mathbf{t}^T\mathbf{x} - \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x}. \tag{5.50}$$

where $\mathbf{t}$ is a column vector denoting the threshold inputs to the neurons, and $\mathbf{W}$ is a symmetric matrix whose entries are given by the weights of the neural net. In view of employing a Hopfield network for minimizing the Lagrangian E of (5.49), the output $\mathbf{x}$ of a Hopfield net is interpreted as the design vector $\mathbf{v}$ associated with a given project, and the weight matrices of the neural net are represented as

$$\mathbf{t} = -\mathbf{c}\,, \text{ and } \mathbf{W} = -\mathbf{Q}\,. \tag{5.51}$$

The Hopfield NN block of the proposed model, as shown in Figure 5.6, is iteratively lowering the energy function $E_H(\mathbf{x})$ of (5.50) as it is described in the following paragraph.

The procedure requires as inputs a CPM network in AoA form, the duration, the resource rate and the early start schedule (ESS) values for each activity from CPM calculations (e.g., earliest start time (EST), latest finish time (LFT) and total float (TF)). Early-start resource curve is also generated using the critical path method (CPM), and shows the resource usage if all the activities were to begin at their earliest dates. The procedure uses only one resource at a time. Multiple resources may be handled sequentially, or by assigning weights to individual resources and adding them up together, as a single resource.

At the beginning of the iterative procedure, the entries of the design vector $\mathbf{v}$ are initialized to certain values corresponding to the early start schedule (ESS) of the given project-network. Let $\mathbf{v}_0$ represents the initial status of $\mathbf{v}$. The Lagrange

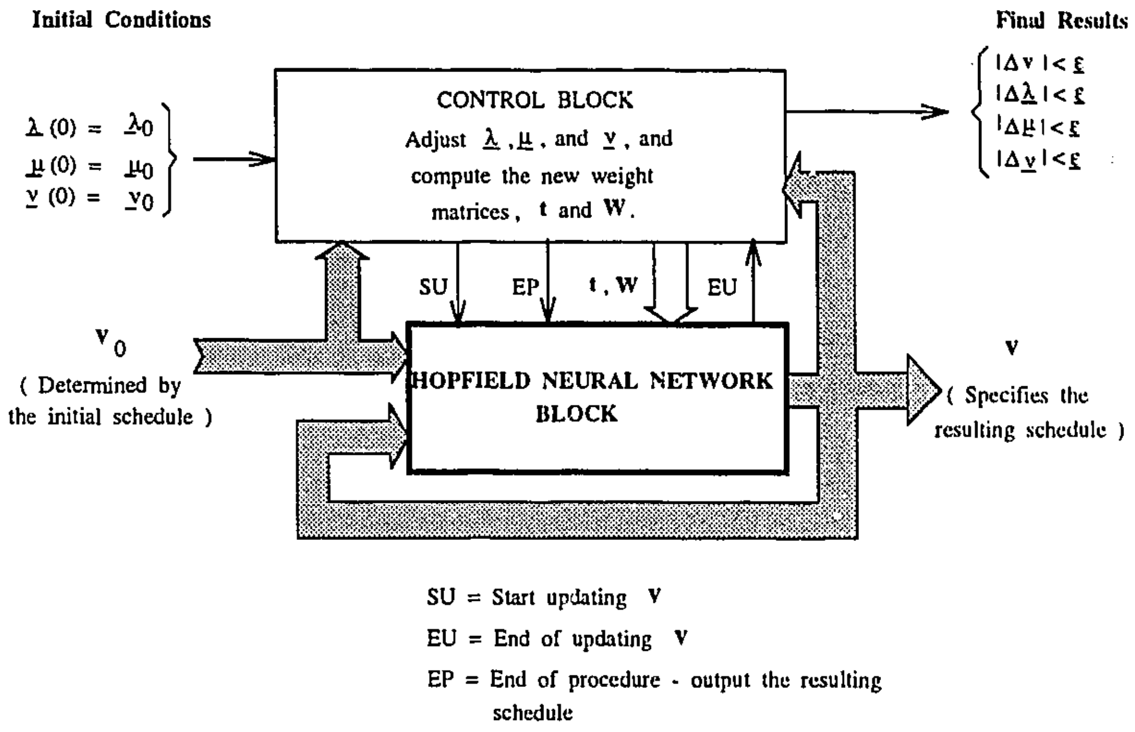

Initial Conditions
Final Results
$\lambda(0) = \lambda_0$
$\mu(0) = \mu_0$
$\nu(0) = \nu_0$
CONTROL BLOCK
Adjust λ, μ, and ν, and compute the new weight matrices, t and W.
$|\Delta v| < \varepsilon$
$|\Delta \lambda| < \varepsilon$
$|\Delta \mu| < \varepsilon$
$|\Delta \nu| < \varepsilon$
SU
EP
t, W
EU
v_0
(Determined by the initial schedule)
HOPFIELD NEURAL NETWORK BLOCK
v
(Specifies the resulting schedule)
SU = Start updating v
EU = End of updating v
EP = End of procedure - output the resulting schedule

Figure 5.6: Proposed neural network model.

multipliers are initialized with some values[3] as described in Chapter 4, i.e.,

$$\underline{\lambda}(0) = \underline{\lambda}_0\,, \quad \underline{\mu}(0) = \underline{\mu}_0\,, \quad \underline{\nu}(0) = \underline{\nu}_0\,. \tag{5.52}$$

The initial set of weights $\mathbf{t}_0$ and $\mathbf{W}_0$ of the Hopfield net are calculated using equations (5.54) and (5.55), for $k = 0$. Once the weight-matrices $\mathbf{t}_0$ and $\mathbf{W}_0$ have been determined, the control block enables the Hopfield neural net to start updating (SU) the design vector. As a result, the Hopfield net proceeds to the adjustment of $\mathbf{v}$ to a new value $\mathbf{v}_1$. Now, according to whether or not the new schedule corresponding to $\mathbf{v}_1$ is still satisfying the constraints (5.29), (5.31), (5.33), two types of actions are possible. Specifically, if $\mathbf{v}_1$ is satisfying the optimization constraints, then the control block allows the Hopfield network to continue in further lowering the energy $\mathrm{E_H}(\mathbf{x})$. On the other hand, if $\mathbf{v}_1$ does not any longer satisfy the constraints (5.29), (5.31) and (5.33), then the control block starts updating the Lagrange parameters to $\underline{\lambda}(1)$, $\underline{\mu}(1)$ and $\underline{\nu}(1)$. Accordingly, the weight matrices of the Hopfield net are adjusted to some new values $\mathbf{t}_1$ and $\mathbf{W}_1$. The proposed neural network model continues to iterate as above, until some ending conditions are met. Specifically, the iterative process is stopped when both the adjustments of the output vector $\mathbf{v}$ as well as the adjustments of Lagrange multipliers become lower than some small positive value ε, i.e.,

$$|\Delta \mathbf{v}| \le \underline{\varepsilon}\,, \quad |\Delta \underline{\lambda}| \le \underline{\varepsilon}\,, \quad |\Delta \underline{\mu}| \le \underline{\varepsilon}\,, \quad |\Delta \underline{\nu}| \le \underline{\varepsilon}\,. \tag{5.53}$$

The decision to end the procedure is made by the control block, as shown in Figure 5.6. At each iteration k, the weight matrices of the Hopfield network are updated by using their functional expressions in terms of the resource requirements, Lagrange multipliers, and penalty terms in the QALM optimization, i.e.,

$$\mathbf{t}_k(i) = -\frac{\partial \mathrm{E}}{\partial v_i} \tag{5.54}$$

[3]Note that in (5.52), $\underline{\lambda}(0)$, $\underline{\mu}(0)$ and $\underline{\nu}(0)$ are column vectors consisting of the Lagrange multipliers λ_j's, μ_j's and ν_i's at the iteration $k = 0$.

$$\mathbf{W}_k(i,j) = -\frac{\partial^2 E}{\partial v_i \partial v_j} \tag{5.55}$$

where E is the augmented Lagrangian of (5.41). Also the Lagrange multipliers at the iteration k are calculated as

$$\lambda_j(k+1) = \lambda_j(k) + \rho_\lambda \cdot g_j^{(1)}(\underline{\mathbf{v}}_k)\,, \tag{5.56}$$

$$\mu_j(k+1) = \mu_j(k) + \rho_\mu \cdot g_j^{(2)}(\underline{\mathbf{v}}_k)\,, \tag{5.57}$$

$$\nu_i(k+1) = \nu_i(k) + \rho_\nu \cdot g_i^{(3)}(\underline{\mathbf{v}}_k)\,, \tag{5.58}$$

for all j's and i's. In (5.56)-(5.58), ρ_λ, ρ_μ and ρ_ν are positive parameters, usually given a value of 1.

5.5 Summary

The development process of a neural network model aimed at solving the resource leveling (RL) problem in construction has been described in this chapter. The model has been developed by mapping a formulation of the RL problem as a quadratic augmented Lagrangian multiplier (QALM) optimization, onto an artificial neural network (ANN) architecture employing a Hopfield-configuration of neural network. It has been shown that the augmented Lagrangian (AL) associated with the RL problem can be interpreted as the energy function of the Hopfield net. The ANN model consists of two main blocks: the Hopfield NN block, and a control block for the computation of the weights of the Hopfield network, and for updating these weights according to the adjustment of Lagrange multipliers in the QALM optimization.

The weights of the Hopfield network can be evaluated by employing their functional expressions in terms of the resource requirements, Lagrange multipliers, and penalty terms in the ALM optimization. However, these functional expressions have to be determined by calculating the gradients and the Hessian of the AL. As a result, the formulation of the weights using their functional expressions is extremely inconvenient for solving the RL problem for different project-networks. Specifically, for each new project-network, the corresponding set of functional expressions of the neuron-weights has to be determined and evaluated, and the control block of the ANN has to be reprogrammed accordingly. In the next chapter, a versatile approach for the derivation of the weight-matrix of the ANN architecture, which can easily accomodate different project-networks, is introduced.

Chapter 6

Formulation of the Weight-Matrix

6.1 Introduction

The key problem facing the mapping of the RL problem as presented in Chapter 5 onto a Hopfield neural network is associated with deriving an efficient technique for determining the matrices **c** and **Q** of (5.49). An in-depth analysis of these matrices reveals that each pair of terms: $\{f_k, \tilde{f}_k\}$, $\{h_j^{(1)}, \tilde{h}_j^{(1)}\}$, $\{h_j^{(2)}, \tilde{h}_j^{(2)}\}$ and $\{h_i^{(3)}, \tilde{h}_i^{(3)}\}$, as well as each term: $\lambda_j g_j^{(1)}$, $\mu_j g_j^{(2)}$, and $\nu_i g_i^{(3)}$, of the right-hand side of (5.41) (with L given by (5.48)), contributes additively to the entries of **c** and **Q** in the form of specific template matrices. An important feature of these templates is that they can be filled-in directly from the early start schedule (ESS) of the given project-diagram, without the need to explicitly derive the augmented Lagrangian in the form expressed by equation (5.41). A methodology for the formulation of each of these template-matrices is introduced in the subsequent sections.

6.2 Template Matrices Corresponding to the Cost Function

Let $\mathcal{S}_k$ represent a set of all the pairs $(j,\tilde{j})$ of different noncritical activities which can be performed during the same day k, i.e.,

$$\mathcal{S}_k \triangleq \left\{ (j,\tilde{j}) \mid j,\tilde{j} \in \mathcal{J}_k \,,\, j < \tilde{j} \right\} . \tag{6.1}$$

Let us also introduce the notation $\mathcal{S}_k^*$ to designate a set which comprises all the pairs of activities (j,j^*), consisting of a noncritical activity j and a critical one j^*, which can be carried out during the same day k, i.e.,

$$\mathcal{S}_k^* \triangleq \{ (j,j^*) \mid j \in \mathcal{J}_k \,,\, j^* \in \mathcal{J}_k^* \} . \tag{6.2}$$

The partial sum $E_k^f(v)$ of the augmented Lagrangian E of (5.41), i.e.,

$$E_k^f(v) \triangleq f_k(v) + \tilde{f}_k(v) , \tag{6.3}$$

can be written in a matrix form as

$$E_k^f(v) = \left[\sum_{(j,j^*)\in\mathcal{S}_k^*} c_{(j,j^*)}^f \right]^T v + \frac{1}{2} v^T \left[\sum_{(j,\tilde{j})\in\mathcal{S}_k} Q_{(j,\tilde{j})}^f \right] v + \mathcal{C}^f , \tag{6.4}$$

where $\mathcal{C}^f$ is a constant, and the template matrices $c_{(j,j^*)}^f$ and $Q_{(j,j^*)}^f$ are given by

$$c_{(j,j^*)}^f \triangleq \begin{array}{c} \begin{array}{ccc} \cdots & \Pi[v_j^{(k)}] & \cdots \end{array} \\ \left[0 \ \ldots \ 0 \ \ \tfrac{1}{2} r_j (r_j + 2 r_{j^*}) \ \ 0 \ \ldots \ 0 \right]^T \end{array} , \tag{6.5}$$

$$Q_{(j,\tilde{j})}^f \triangleq \begin{array}{c|ccccc|} & \cdots & \Pi[v_j^{(k)}] & \cdots & \Pi[v_{\tilde{j}}^{(k)}] & \cdots \\ \vdots & & & & & \\ \Pi[v_j^{(k)}] & & 0 & & r_j r_{\tilde{j}} & \\ \vdots & & & & & \\ \Pi[v_{\tilde{j}}^{(k)}] & & r_{\tilde{j}} r_j & & 0 & \\ \vdots & & & & & \end{array} . \tag{6.6}$$

The operator $\Pi[v_j^{(k)}]$ used in (6.5) and (6.6), is assumed to return the row-number of the element $v_j^{(k)}$ of the design vector $\mathbf{v}$. This is best explained in the following example.

Example 6.1 As an example, let us consider the project-diagram of Figure 6.1, for which the ESS is depicted in Figure 6.2. In this example, the design vector $\mathbf{v}$ is given by

$$\mathbf{v} = \begin{bmatrix} v_1^{(1)} & v_1^{(2)} & v_1^{(3)} & v_1^{(4)} & v_1^{(5)} & v_1^{(6)} & v_2^{(1)} & v_2^{(2)} & v_2^{(3)} \end{bmatrix}^T , \tag{6.7}$$

while the sets $\mathcal{S}_1$ and $\mathcal{S}_1^*$ consist of the following elements:

$$\mathcal{S}_1 = \{(1,2)\} , \text{ and } \mathcal{S}_1^* = \{(1,3),(2,3)\} . \tag{6.8}$$

Consequently, corresponding to the day k = 1, we get

$$\mathbf{c}^f_{(1,3)} = \begin{bmatrix} \frac{1}{2}r_1(r_1+2r_3) & 0 & \dots & 0 \end{bmatrix}^T , \tag{6.9}$$

$$\mathbf{c}^f_{(2,3)} = \begin{bmatrix} 0 & \dots & 0 & \frac{1}{2}r_2(r_2+2r_3) & 0 & 0 \end{bmatrix}^T , \tag{6.10}$$

$$\mathbf{Q}^f_{(1,2)} = \begin{bmatrix} 0 & \dots & 0 & r_1r_2 & 0 & 0 \\ \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & 0 & 0 \\ r_2r_1 & \dots & 0 & 0 & 0 & 0 \\ 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \dots & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{6.11}$$

The expressions (6.9)-(6.11) can be verified by observing that

$$f_1(\mathbf{v}) = \frac{1}{2}\left(r_1v_1^{(1)} + r_2v_2^{(1)} + r_3\right)^2 , \tag{6.12}$$

$$\tilde{f}_1(\mathbf{v}) = \frac{1}{2}\left(r_1^2\tilde{v}_1^{(1)} + r_2^2\tilde{v}_2^{(1)}\right) , \tag{6.13}$$

and therefore

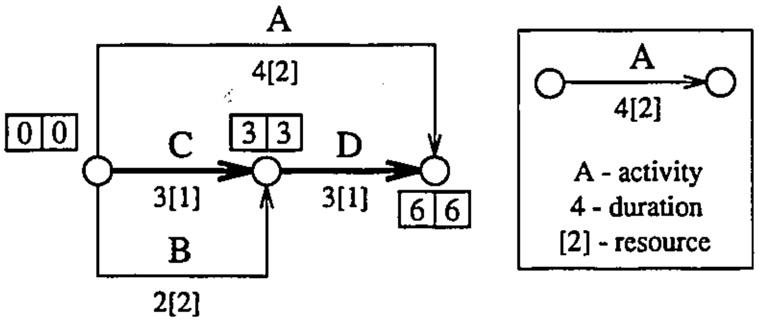


Figure 6.1: Activity-on-arrow diagram of the project used in Example 6.1.

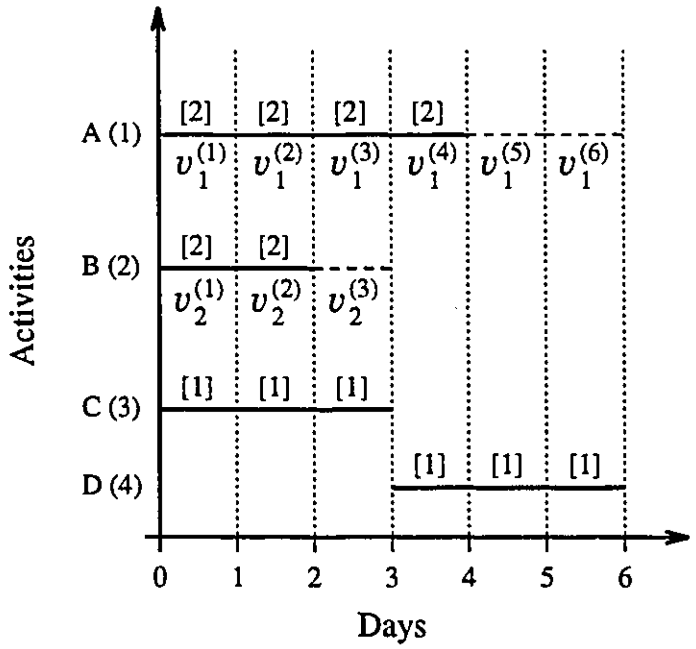


Figure 6.2: Early start schedule for the project-network of Figure 6.1.

$$E_1^f = f_1 + \tilde{f}_1 =$$

$$= \left[(\frac{1}{2}r_1^2 + r_1 r_3)\, v_1^{(1)} + (\frac{1}{2}r_2^2 + r_2 r_3)\, v_2^{(1)} \right] + \left[\frac{1}{2} \left(2 r_1 r_2\, v_1^{(1)} v_2^{(1)} \right) \right] + r_3^2 . \quad (6.14)$$

Note however, that matrices (6.9)-(6.11) have been filled-in directly using the ESS of Figure 2, and without carying out the calculations of (6.12)-(6.14).

6.3 Template Matrices Corresponding to the Constraints Imposing the Duration of Activities

Let $\mathcal{T}_j$ denote a set consisting of all the pairs of different days $(k, \tilde{k})$, during which the activity j can take place, i.e,

$$\mathcal{T}_j \triangleq \left\{ (k, \tilde{k}) \mid k, \tilde{k} \in \{K_1(j), \ldots, K_2(j)\} ,\ k < \tilde{k} \right\} . \quad (6.15)$$

Each partial sum E_j^{h1}, $j = 1, 2, \ldots, J$, of the augmented Lagrangian E of (5.41) (with L given by (5.48)), with

$$E_j^{h1}(v) \triangleq h_j^{(1)}(v) + \tilde{h}_j^{(1)}(v) , \quad (6.16)$$

can be written in a matrix form as

$$E_j^{h1}(v) = \left[\sum_{k=K_1(j)}^{K_2(j)} \mathbf{c}_k^{h1} \right]^T v \;+\; \frac{1}{2} v^T \left[\sum_{(k,\tilde{k}) \in \mathcal{T}_j} \mathbf{Q}_{(k,\tilde{k})}^{h1} \right] v + \mathcal{C}^{h1} , \quad (6.17)$$

where $\mathcal{C}^{h1}$ is a constant, and the template matrices $\mathbf{c}_k^{h1}$ and $\mathbf{Q}_{(k,\tilde{k})}^{h1}$ are given by

$$\mathbf{c}_k^{h1} \triangleq \begin{array}{ccccccc} & & \ldots & \Pi[v_j^{(k)}] & \ldots & & \\ [0 & \ldots & 0 & \alpha_j(\frac{1}{2} - d_j) & 0 & \ldots & 0]^T \end{array} , \quad (6.18)$$

$$\mathbf{Q}_{(k,\tilde{k})}^{h1} \triangleq \begin{array}{c|ccccc|} & \ldots & \Pi[v_j^{(k)}] & \ldots & \Pi[v_j^{(\tilde{k})}] & \ldots \\ \hline \vdots & & & & & \\ \Pi[v_j^{(k)}] & & 0 & & \alpha_j & \\ \vdots & & & & & \\ \Pi[v_j^{(\tilde{k})}] & & \alpha_j & & 0 & \\ \vdots & & & & & \end{array} . \quad (6.19)$$

Example 6.2 Referring back to the project-diagram of Figure 6.1, the matrices $\sum_{k=K_1(2)}^{K_2(2)} \mathbf{c}_k^{h1}$ and $\sum_{(k,\bar{k})\in\mathcal{T}_2} \mathbf{Q}_{(k,\bar{k})}^{h1}$, corresponding to the activity j = 2, are filled-in directly by using the ESS of Figure 6.2 as follows. It is observed that $\Pi[v_2^{(K_1(2))}] = 7$, $\Pi[v_2^{(K_2(2))}] = 9$, and $d_2 = 2$ (i.e., $\alpha_2(\frac{1}{2} - d_2) = -\frac{3}{2}\alpha_2$), and therefore

$$\sum_{k=K_1(2)}^{K_2(2)} \mathbf{c}_k^{h1} = -\frac{3}{2}\alpha_2 \begin{bmatrix} 0 & \dots & 0 & 1 & 1 & 1 \end{bmatrix}^T , \tag{6.20}$$

$$\sum_{(k,\bar{k})\in\mathcal{T}_2} \mathbf{Q}_{(k,\bar{k})}^{h1} = \begin{bmatrix} 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \dots & 0 & 0 & \alpha_2 & \alpha_2 \\ 0 & \dots & 0 & \alpha_2 & 0 & \alpha_2 \\ 0 & \dots & 0 & \alpha_2 & \alpha_2 & 0 \end{bmatrix} . \tag{6.21}$$

The structure of matrices (6.20) and (6.21) can be verified by observing that $h_2^{(1)}$ and $\tilde{h}_2^{(1)}$ are given by

$$h_2^{(1)}(\mathbf{v}) = \frac{1}{2}\alpha_2\left(v_2^{(1)} + v_2^{(2)} + v_2^{(3)} - d_2\right)^2 , \tag{6.22}$$

$$\tilde{h}_2^{(1)} = \frac{1}{2}\alpha_2\left(\tilde{v}_2^{(1)} + \tilde{v}_2^{(2)} + \tilde{v}_2^{(3)}\right) . \tag{6.23}$$

It is to be noted that each term $\lambda_j g_j^{(1)}$ of the summation $\sum_{j=1}^{J} \lambda_j g_j^{(1)}$ of the L-part of (5.41) contributes additively to the entries of the column vector **c** of (5.49), in the form of a template matrix

$$\begin{array}{cccccccccc} & & & & \dots & \Pi[v_j^{(k)}] & \dots & & & \\ \mathbf{c}_j^{g1} \triangleq & [\,0 & \dots & 0 & & \lambda_j & & 0 & \dots & 0\,]^T \end{array} . \tag{6.24}$$

6.4 Template Matrices Corresponding to the Constraints Imposing the Continuity of Activities

As in the previous two sections, it is convenient to introduce here several set-notations. Specifically, let

$$\tilde{\mathcal{U}}_j \triangleq \{(k,\tilde{k}) \,|\, k,\tilde{k} \in \{K_1(j),\ldots,K_3(j)-1\}, k<\tilde{k}\}\,, \tag{6.24}$$

$$\mathcal{U}_j^* \triangleq \{(k,k^*) \,|\, k,k^* \in \{K_3(j),\ldots,K_2(j)\}, k<k^*\}\,, \tag{6.25}$$

$$\tilde{\mathcal{U}}_j^* \triangleq \{\, (\tilde{k},k^*) \;|\; \tilde{k} \in \{K_1(j),\ldots,K_3(j)-1\}\,,\quad k^* \in \{K_3(j),\ldots,K_2(j)\}\,\}\,. \tag{6.26}$$

Each partial sum E_j^{h2}, $j = 1,2,\ldots,J$, with

$$E_j^{h2}(v) \triangleq h_j^{(2)}(v) + \tilde{h}_j^{(2)}(v)\;, \tag{6.27}$$

can be written in a matrix form as

$$E_j^{h2}(v) = [c_j^{h2}]^T\, v \,+\, \frac{1}{2} v^T\, [Q_j^{h2}]\, v \,+\, \mathcal{C}^{h2}\;, \tag{6.28}$$

where

$$c_j^{h2} \triangleq \sum_{k=K_1(j)}^{K_3(j)-1} c_k^{h2} \,+\, \sum_{\tilde{k}=K_3(j)}^{K_2(j)} c_{\tilde{k}}^{h2}\;, \tag{6.29}$$

$$Q_j^{h2} \triangleq \sum_{(k,\tilde{k})\in\tilde{\mathcal{U}}_j} Q_{(k,\tilde{k})}^{h2} \,+\, \sum_{(k,k^*)\in\mathcal{U}_j^*} Q_{(k,k^*)}^{h2} \,+\, \sum_{(\tilde{k},k^*)\in\tilde{\mathcal{U}}_j^*} Q_{(\tilde{k},k^*)}^{h2}\;, \tag{6.30}$$

and $\mathcal{C}^{h2}$ is a constant. The template matrices c_k^{h2} and $c_{\tilde{k}}^{h2}$ of (6.29) are given by

$$c_k^{h2} \triangleq \begin{array}{c} \begin{matrix} \ldots & \Pi[v_j^{(k)}] & \ldots \end{matrix} \\ [\ldots\; 0\;\; \beta_j(k+d_j)[\tfrac{1}{2}(k+d_j)-\delta_j]\;\; 0\; \ldots]^T \end{array}, \tag{6.31}$$

$$c_{\tilde{k}}^{h2} \triangleq \begin{array}{c} \begin{matrix} \ldots & \Pi[v_j^{(\tilde{k})}] & \ldots \end{matrix} \\ [\;\ldots\;\; 0\;\;\; \beta_j\tilde{k}(\tfrac{1}{2}\tilde{k}-\delta_j)\;\;\; 0\;\;\ldots\;]^T \end{array}, \tag{6.32}$$

while the template matrices $\mathbf{Q}^{h2}_{(\mathrm{k},\tilde{\mathrm{k}})}$, $\mathbf{Q}^{h2}_{(\mathrm{k},\mathrm{k}^*)}$, and $\mathbf{Q}^{h2}_{(\tilde{\mathrm{k}},\mathrm{k}^*)}$ of (6.30) are defined as

$$\mathbf{Q}^{h2}_{(\mathrm{k},\tilde{\mathrm{k}})} \triangleq \beta_{\mathrm{j}} \begin{array}{c|ccccc|} & \cdots & \Pi[v_{\mathrm{j}}^{(\mathrm{k})}] & \cdots & \Pi[v_{\mathrm{j}}^{(\tilde{\mathrm{k}})}] & \cdots \\ \vdots & & & & & \\ \Pi[v_{\mathrm{j}}^{(\mathrm{k})}] & & 0 & & (\mathrm{k}+d_{\mathrm{j}})(\tilde{\mathrm{k}}+d_{\mathrm{j}}) & \\ \vdots & & & & & \\ \Pi[v_{\mathrm{j}}^{(\tilde{\mathrm{k}})}] & & (\tilde{\mathrm{k}}+d_{\mathrm{j}})(\mathrm{k}+d_{\mathrm{j}}) & & 0 & \\ \vdots & & & & & \end{array}, \tag{6.33}$$

$$\mathbf{Q}^{h2}_{(\mathrm{k},\mathrm{k}^*)} \triangleq \beta_{\mathrm{j}} \begin{array}{c|ccccc|} & \cdots & \Pi[v_{\mathrm{j}}^{(\mathrm{k})}] & \cdots & \Pi[v_{\mathrm{j}}^{(\mathrm{k}^*)}] & \cdots \\ \vdots & & & & & \\ \Pi[v_{\mathrm{j}}^{(\mathrm{k})}] & & 0 & & \mathrm{k}\mathrm{k}^* & \\ \vdots & & & & & \\ \Pi[v_{\mathrm{j}}^{(\mathrm{k}^*)}] & & \mathrm{k}^*\mathrm{k} & & 0 & \\ \vdots & & & & & \end{array}, \tag{6.34}$$

$$\mathbf{Q}^{h2}_{(\tilde{\mathrm{k}},\mathrm{k}^*)} \triangleq \beta_{\mathrm{j}} \begin{array}{c|ccccc|} & \cdots & \Pi[v_{\mathrm{j}}^{(\tilde{\mathrm{k}})}] & \cdots & \Pi[v_{\mathrm{j}}^{(\mathrm{k}^*)}] & \cdots \\ \vdots & & & & & \\ \Pi[v_{\mathrm{j}}^{(\tilde{\mathrm{k}})}] & & 0 & & (\tilde{\mathrm{k}}+d_{\mathrm{j}})\mathrm{k}^* & \\ \vdots & & & & & \\ \Pi[v_{\mathrm{j}}^{(\mathrm{k}^*)}] & & \mathrm{k}^*(\tilde{\mathrm{k}}+d_{\mathrm{j}}) & & 0 & \\ \vdots & & & & & \end{array}. \tag{6.35}$$

Example 6.25 Referring back to the project-diagram of Figure 6.1, and considering activity j = 2 of the project we have $\mathrm{K}_1(2) = 1$, $\mathrm{K}_2(2) = 3$, $d_2 = 2$, $\delta_2 = 7$, $\tilde{\mathcal{U}}_2 = \{(1,2)\}$, $\mathcal{U}_2^* = \emptyset$, and $\tilde{\mathcal{U}}_2^* = \{(1,3),(2,3)\}$. Therefore,

$$\mathbf{c}_2^{h2} = \sum_{\mathrm{k}=1}^{2} \mathbf{c}_{\mathrm{k}}^{h2} + \sum_{\tilde{\mathrm{k}}=3}^{3} \mathbf{c}_{\tilde{\mathrm{k}}}^{h2} = \beta_2 \begin{bmatrix} 0 & \cdots & 0 & -\frac{33}{2} & -20 & -\frac{33}{2} \end{bmatrix}^{\mathrm{T}}, \tag{6.36}$$

$$\mathbf{Q}_2^{h2} = \beta_2 \begin{bmatrix} 0 & \ldots & 0 & 0 & 0 & 0 \\ \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 3\cdot 4 & 3\cdot 3 \\ 0 & \ldots & 0 & 4\cdot 3 & 0 & 3\cdot 4 \\ 0 & \ldots & 0 & 3\cdot 3 & 3\cdot 4 & 0 \end{bmatrix} . \tag{6.37}$$

The structure of matrices (6.36) and (6.37) can be easily verified by observing that $h_2^{(2)}$ and $\tilde{h}_2^{(2)}$ are given by

$$h_2^{(2)} = \frac{1}{2}\beta_2\left(3v_2^{(1)} + 4v_2^{(2)} + 3v_2^{(3)} - 7\right)^2 , \tag{6.38}$$

$$\tilde{h}_2^{(2)} = \frac{1}{2}\beta_2\left(3^2\tilde{v}_2^{(1)} + 4^2\tilde{v}_2^{(2)} + 3^2\tilde{v}_2^{(3)}\right) . \tag{6.39}$$

It can be also observed that each term $\mu_j g_j^{(2)}$ of the summation $\sum_{j=1}^{J} \mu_j g_j^{(2)}$ of the L-part of (5.41) contributes additively to the entries of the column vector **c** of (5.49), in the form of two template matrices

$$\begin{array}{rccccccccc} & & & \ldots & \Pi[v_j^{(k)}] & \ldots & & & \\ \mathbf{c}_j^{g2}(\mathrm{k}) \triangleq & [\,0 & \ldots & 0 & \mu_j\frac{1}{2}(\mathrm{k}+d_j) & 0 & \ldots & 0\,]^{\mathrm{T}} \end{array} \tag{6.40}$$

$$\begin{array}{rccccccccc} & & & \ldots & \Pi[v_j^{(\tilde{k})}] & \ldots & & & \\ \mathbf{c}_j^{g2}(\tilde{\mathrm{k}}) \triangleq & [\,0 & \ldots & 0 & \mu_j\frac{1}{2}\tilde{\mathrm{k}} & 0 & \ldots & 0\,]^{\mathrm{T}}. \end{array} \tag{6.41}$$

6.5 Template Matrices Corresponding to the Constraints Imposing the Precedence of Activities

In the case of the constraints regarding the precedence of activities, it can be noted that each term E_i^{h3}, $i = 1, 2, \ldots, I$, of the AL of (5.41), with

$$E_i^{h3}(v) \triangleq h_i^{(3)}(v) + \tilde{h}_i^{(3)}(v) = \frac{1}{2}\gamma_i v_p^{(k)} v_j^{(k)} , \tag{6.42}$$

can be written in a matrix form as

$$E^{h3}(v) = \sum_{i=1}^{I} E_i^{h3}(v) = \frac{1}{2} v^T \left[\sum_{i=1}^{I} \mathbf{Q}_i^{h3} \right] v + C^{h3} , \tag{6.43}$$

where the template matrices $\mathbf{Q}_i^{h3}$ are given by

$$\mathbf{Q}_i^{h3} = \frac{1}{2}\gamma_i \mathbf{q}_i^{h3} , \tag{6.44}$$

with

$$\mathbf{q}_i^{h3} \triangleq \begin{array}{c|ccccc|} & \cdots & \Pi[v_p^{(k)}] & \cdots & \Pi[v_j^{(k)}] & \cdots \\ \hline \vdots & & & & & \\ \Pi[v_p^{(k)}] & & 0 & & 1 & \\ \vdots & & & & & \\ \Pi[v_j^{(k)}] & & 1 & & 0 & \\ \vdots & & & & & \end{array} , \tag{6.45}$$

and C^{h3} is a constant.

It can also be observed that each term $\nu_i g_i^{(3)}$ of the summation $\sum_{i=1}^{I} \nu_i g_i^{(3)}$ of the L-part of (5.41), contributes additively to the entries of matrix $\mathbf{Q}$ of (5.49) in the form of a template matrix

$$\mathbf{Q}_i^{g3} = \frac{1}{2}\nu_i \mathbf{q}_i^{h3} . \tag{6.46}$$

6.6 Summary

A methodology for the derivation of the weight-matrix of a neural network for resource leveling has been introduced and described in this chapter. An in-depth study of the matrices **c** and **Q** arising from the formulation of an RL problem as a quadratic augmented Lagrangian optimization has revealed some very useful structural properties. They have been formalized as template-matrix contributions of different terms of the AL associated with the RL problem, to the entries of the matrices **c** and **Q**. In the case of a specific project-diagram, several examples illustrating the structure of the template matrices have been provided. It has been emphasized that the template matrices can be easily filled-in easily, using the ESS of a project-diagram.

The theoretical framework described in this section allows for the derivation of a computationally-efficient yet versatile algorithm, which can easily accomodate different structures of project-networks, for the computation of the weight-matrix of a Hopfield-based NN architecture for resource leveling. With such an algorithm, the weight-matrix is filled-in directly using characteristic templates, without a need to explicitly determine the functional expressions of the weights by computing the gradients and the Hessian of the AL.

Chapter 7

Model Validation

7.1 Introduction

The objective of this chapter is to validate the effectiveness of the proposed NN model. The IEEE (1983) defines validation as "...the process of evaluating software at the end of the development process to ensure compliance with software requirements". Verification is an area of validation, with the distinction being that verification is concerned with whether the software, or the system, operates corectly, and validation is concerned with whether the system is correct or appropriate for the problem to be solved. The IEEE defines verification as "... the process of determining whether the products of a given phase of software development meets all the requirements established during the previous phase".

The proposed NN model has been validated using three case examples, which are presented next.

7.2 Case Example 1

Example 7.1 The project-diagram for the first example consists of five activities (J=5), A, B, C, D, and E, and the critical path runs along activities C, D, and E, as shown in Figure 7.1(a). The activities that comprise the project are represented by arrows. The arrow tail and arrow head represent the start and finish of an activity, respectively. The bar-chart, or the Early Start Schedule (ESS) for the described project is shown in Figure 7.1(b). The project duration K=3 days.

The dashed line, following an activity, as illustrated in Figure 7.1(b), corresponds to the activity's total float. For simplicity, the position of an activity on the bar-chart is denoted only by one index, written alongside activity's symbol, as illustrated in Figure 7.1(b). For example, C(3) indicates that activity C is listed the third on the bar-chart. As a general rule, the noncritical activities, for which the output variables exist, are listed first, followed by the critical activities. In this example, there are two noncritical activities (J=2), A and B, each having a float of one day, and three critical activities, C, D, and E.

The start and finishing times for activity A(1) are : $K_1(1) = 1$, and $K_2(1) = 2$. Similarly, the start and finishing times for activity B(2) are: $K_1(2) = 2$, and $K_2(2) = 3$.

Corresponding to noncritical activities, the design vector, defined in (5.42), is equal to

$$\mathbf{v} = [\,\mathbf{v}_1\,\mathbf{v}_2\,]^{\mathrm{T}},$$

where

$$\mathbf{v}_1 = \left[\, v_1^{(1)}\, v_1^{(2)} \,\right]$$

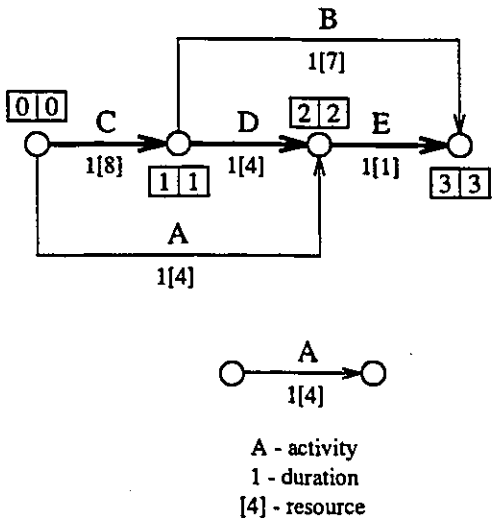
corresponds to activity A(1), and

$$\mathbf{v}_2 = \left[\, v_2^{(2)}\, v_2^{(3)} \,\right]$$

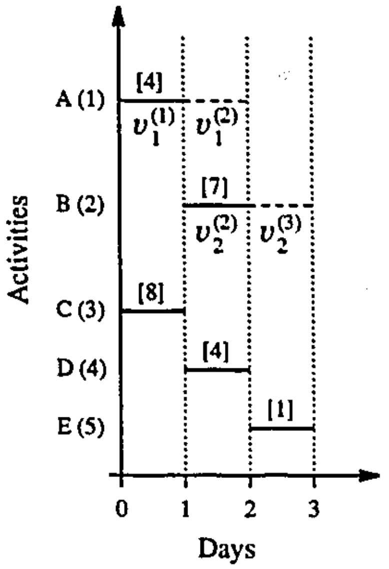
corresponds to activity B(2). Thus,

$$\mathbf{v} = \left[\, v_1^{(1)}\, v_1^{(2)}\, v_2^{(2)}\, v_2^{(3)} \,\right]^{\mathrm{T}}.$$

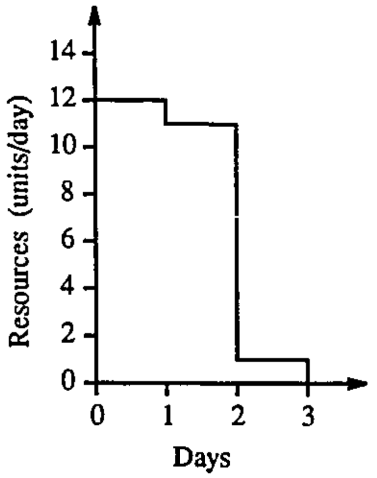
The design vector could be initialized based on ESS, or with arbitrary values. For this case example, all the output variables have been initially assigned the arbitrary



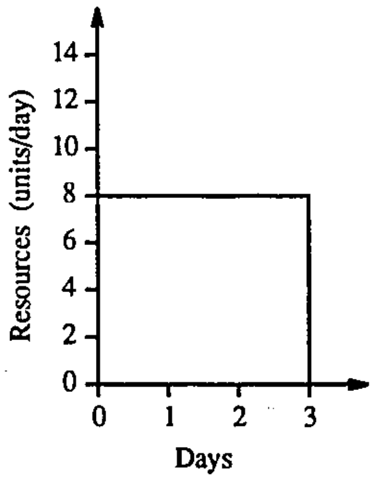
(a) Activity-on-arrow diagram



(b) Early start schedule



(c) Histogram of required resources before leveling



(d) Histogram of required resources after leveling

Figure 7.1: Resource leveling for Case Example 1

value of $\frac{1}{2}$, i.e.,

$$\mathbf{v}_{\text{initial}} = \left[v_1^{(1)} = \frac{1}{2} \; , \quad v_1^{(2)} = \frac{1}{2} \; , \quad v_2^{(2)} = \frac{1}{2} \; , \quad v_2^{(3)} = \frac{1}{2} \right]^{\mathrm{T}} .$$

By summing the resources required by the activities which occur on any given day, one may represent the project required resources as a function of time, as shown in Figure 7.1(c). In this example, only one type of resource is considered. It is to be noted here that the amount of required resources varies over the project duration, and therefore there is a need for RL.

The RL procedure, illustrated in Figure 5.6, requires as inputs the following:

- activity values: duration, resource rate (from the project's AoA diagram), ESS values, e.g., EST, LFT and TF (from the CPM calculations)
- the relationships among activities (from the project's AoA diagram)
- the project duration (from the CPM calculations)
- the penalty parameters (assumed 1)
- an acceptable value for the NN error, ϵ (assumed 10^{-2})
- initial values for the output variables, denoted by $\mathbf{v}_0$ (either values corresponding to the ESS, or arbitrary values)
- initial values for Lagrange parameters

The initial values for Lagrange parameters are determined following the procedure described in Section 4.3. The total number of constraints is equal to two, since there are only two critical activities for which the constraints regarding their durations have to be imposed. In this example, there is no need to impose constraints regarding the continuity (i.e., all the durations are equal to 1) nor the precedence of the noncritical activities (i.e., there is no noncritical activity which precedes another noncritical activity). The first type of constraints specifies that the duration of a noncritical activity has to be equal to the number of active neurons for that activity at any moment in time. For noncritical activities A and B the two constraints are:

$$v_1^{(1)} + v_1^{(2)} = 1 \quad \text{and} \quad v_2^{(2)} + v_2^{(3)} = 1$$

or using a similar matrix notation as in expression (4.10), the constraints can be written as:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} v_1^{(1)} \\ v_1^{(2)} \\ v_2^{(2)} \\ v_2^{(3)} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

where the constraint-matrix **A** is:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} .$$

In order to find an estimate of the Lagrange multipliers, first the constraint-matrix **A** is partitioned. Specifically, for each noncritical activity one variable is selected from the design vector, i.e., $v_1^{(1)}$ for activity A(1), and $v_2^{(2)}$ for activity B(2). Then, the columns corresponding to $v_1^{(1)}$ and $v_2^{(2)}$ from the constraint-matrix are selected, and the non-singular decomposition **V** is equal to:

$$\mathbf{V} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} .$$

In order to solve the linear system described by (4.19), matrices $\mathbf{Q}_{0f}$ and $\mathbf{c}_{0f}$ have to be partitioned accordingly. The derivation of the matrices $\mathbf{Q}_{0f}$ and $\mathbf{c}_{0f}$ is based on the objective function which, for this example, is:

$$f(\mathbf{v}) = \frac{1}{2}\sum_{k=1}^{3}\left(\sum_{j\in\mathcal{J}_k} r_j v_j^{(k)} + \sum_{j\in\mathcal{J}_k^*} r_j\right)^2 ,$$

where $\mathcal{J}_1 = \{1\}$, $\mathcal{J}_1^* = \{3\}$, $\mathcal{J}_2 = \{1,2\}$, $\mathcal{J}_2^* = \{4\}$, $\mathcal{J}_3 = \{2\}$, $\mathcal{J}_3^* = \{5\}$. The resource requirements, as shown in Figure 7.1(a) and (b) are: $r_1 = 4$, $r_2 = 7$, $r_3 = 8$,

$r_4 = 4$ and $r_5 = 1$. Substituting the values for the resource requirements r_j, and for the sets $\mathcal{J}_k$ and $\mathcal{J}_k^*$, the cost (objective) function becomes:

$$f(\mathrm{v}) = \frac{1}{2}[\left(r_1 v_1^{(1)} + r_3\right)^2 + \left(r_1 v_1^{(2)} + r_2 v_2^{(2)} + r_4\right)^2 + \left(r_2 v_2^{(3)} + r_5\right)^2] =$$

$$= \frac{1}{2}[r_1^2 v_1^{(1)} v_1^{(1)} + r_1^2 v_1^{(2)} v_1^{(2)} + r_2^2 v_2^{(2)} v_2^{(2)} + r_2^2 v_2^{(3)} v_2^{(3)} + 2 r_1 r_2 v_1^{(2)} v_2^{(2)} +$$

$$+2\left(r_1 r_3 v_1^{(1)} + r_1 r_4 v_1^{(2)} + r_2 r_4 v_2^{(2)} + r_2 r_5 v_2^{(3)}\right) + r_3^2 + r_4^2 + r_5^2]$$

From the cost function, matrices $\mathbf{Q}_{0f}$ and $\mathbf{c}_{0f}$ are calculated as follows:

$$\mathbf{Q}_{0f} = \begin{bmatrix} r_1^2 & 0 & 0 & 0 \\ 0 & r_1^2 & r_1 r_2 & 0 \\ 0 & r_1 r_2 & r_2^2 & 0 \\ 0 & 0 & 0 & r_2^2 \end{bmatrix} = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 16 & 28 & 0 \\ 0 & 28 & 49 & 0 \\ 0 & 0 & 0 & 49 \end{bmatrix}$$

and

$$\mathbf{c}_{0f} = [\,2r_1r_3 \;\; 2r_1r_4 \;\; 2r_2r_4 \;\; 2r_2r_5\,]^{\mathrm{T}} = [\,64 \;\; 32 \;\; 56 \;\; 14\,]^{\mathrm{T}}.$$

Then, the columns corresponding to $\mathrm{v}_1^{(1)}$ and $\mathrm{v}_2^{(2)}$ are selected, and the non-singular decompositions $\mathbf{Q}_{0f}^*$ and $\mathbf{c}_{0f}^*$ are:

$$\mathbf{Q}_{0f}^* = \begin{bmatrix} 16 & 0 \\ 0 & 49 \end{bmatrix},$$

and

$$\mathbf{c}_{0f}^* = [\,64 \;\; 56\,]^{\mathrm{T}}.$$

Finally, the linear system described by (4.19) can be solved,

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = -\left(\begin{bmatrix} 64 \\ 56 \end{bmatrix} + \begin{bmatrix} 16 & 0 \\ 0 & 49 \end{bmatrix} \begin{bmatrix} v_1^{(1)}(0) \\ v_2^{(2)}(0) \end{bmatrix}\right),$$

and its solutions are:

$$\lambda_1 = -\left(64 + 16 v_1^{(1)}(0)\right) = -64$$

$$\lambda_2 = -\left(56 + 49 v_2^{(2)}(0)\right) = -105$$

for $v_1^{(1)}(0) = 0$ and $v_2^{(2)}(0) = 1$.

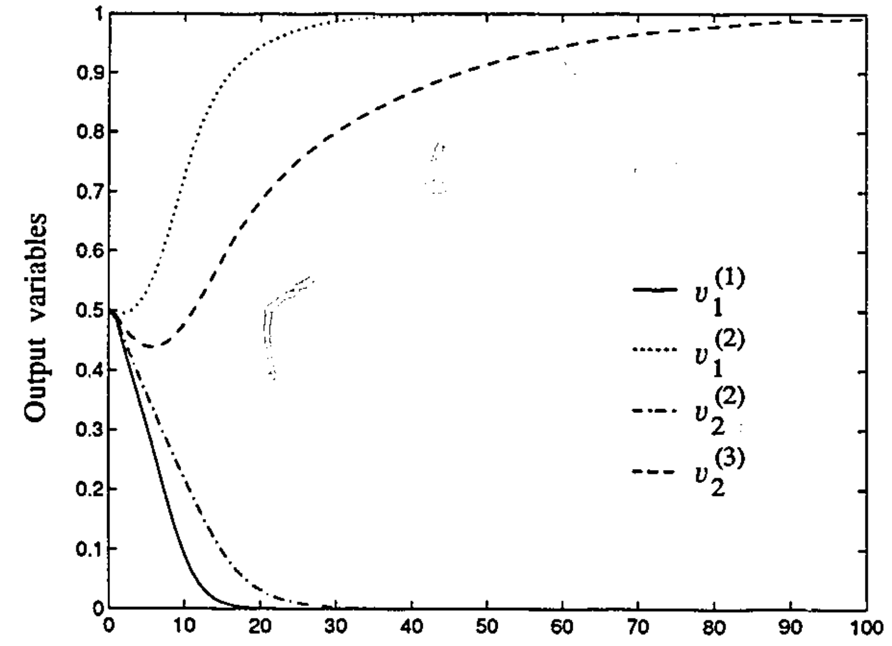
The initialization step is done after the weight matrices $\mathbf{c}_0$ and $\mathbf{Q}_0$ are filled-in directly from the ESS of the project, as discussed in Chapter 6. Having determined the initial weights of the Hopfield NN block, the proposed ANN scheme shown in Figure 5.6 performs the leveling process until the ending conditions are met. The convergence of the output variables is illustrated in Figure 7.2(a). Variables $v_1^{(1)}$ and $v_2^{(2)}$ have converged to zero, while variables $v_1^{(2)}$ and $v_2^{(3)}$ have converged to one. This means that, after leveling, the start time of activity A has been shifted to day 2 (i.e., $v_1^{(2)} = 1$) , and the start time of activity B has been shifted to day 3 (i.e., $v_2^{(3)} = 1$). The final values of the design vector $\mathbf{v}$ represent the output of the the RL procedure, i.e.,

$$\mathbf{v}_{\text{final}} = \left[v_1^{(1)} = 0 \ , \quad v_1^{(2)} = 1 \ , \quad v_2^{(2)} = 0 \ , \quad v_2^{(3)} = 1 \right]^T .$$

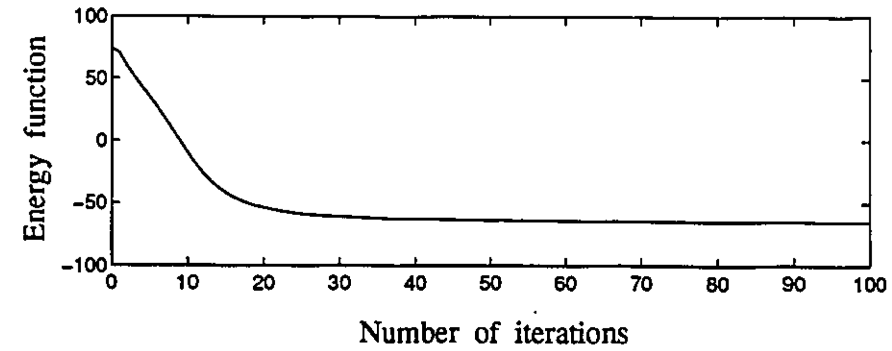
The value of the energy function decreased from -12 for the initial schedule, to -60 for the levelled schedule, as shown in Figure 7.2(b). The implementation of the ANN model was carried out using Matlab. The model reached a steady state after 100 iterations, consuming 20 seconds of computer time. In this example case, the proposed ANN model provided an optimal solution, as illustrated in Figure 7.1(d). As it can be seen from the after leveling resource histogram, the fluctuations in the pattern of resource usage have been eliminated.

7.3 Case Example 2

Example 7.2 As the second example let us consider the project-diagram of Figure 7.3(a), for which the ESS is depicted in Figure 7.3(b), and the histogram of required resources before leveling is shown in Figure 7.3(c).The project consists of four activities (J=4), A, B, C, and D, and the critical path runs along activities C and D.The project duration is K=6 days. In this example activities C and D are critical, while activities A and B are noncritical, as illustrated in Figure 7.3(b).



(a) Convergence of the output variables



(b) Convergence of the energy function

Figure 7.2: An illustration of the convergence of the proposed NN for Example 1

The start and finishing times of activity A(1) are : $K_1(1) = 1$, and $K_2(1) = 6$. The total float of activity A is two days. Similarly, the start and finishing times for activity B(2) are: $K_1(2) = 1$, and $K_2(2) = 3$. The total float for activity B is one day. Corresponding to noncritical activities, the design vector, defined in (5.42), is equal to

$$\mathbf{v} = \left[\, v_1^{(1)}\; v_1^{(2)}\; v_1^{(3)}\; v_1^{(4)}\; v_1^{(5)}\; v_1^{(6)}\; v_2^{(1)}\; v_2^{(2)}\; v_2^{(3)} \,\right]^T.$$

where

$$\left[\, v_1^{(1)}\; v_1^{(2)}\; v_1^{(3)}\; v_1^{(4)}\; v_1^{(5)}\; v_1^{(6)} \,\right]$$

corresponds to activity A(1), and

$$\left[\, v_2^{(1)}\; v_2^{(2)}\; v_2^{(3)} \,\right]$$

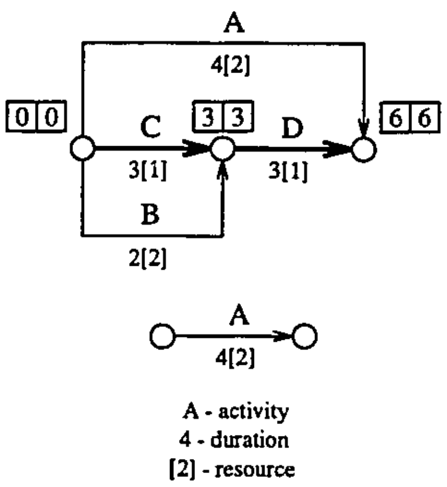
corresponds to activity B(2). As in the previous case example, all the output variables have been initially assigned the arbitrary value of $\frac{1}{2}$. After leveling, their values have changed as illustrated in Figure 7.4(a).

$$\mathbf{v}_{\mathsf{final}} = [\,0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0\,]^T$$

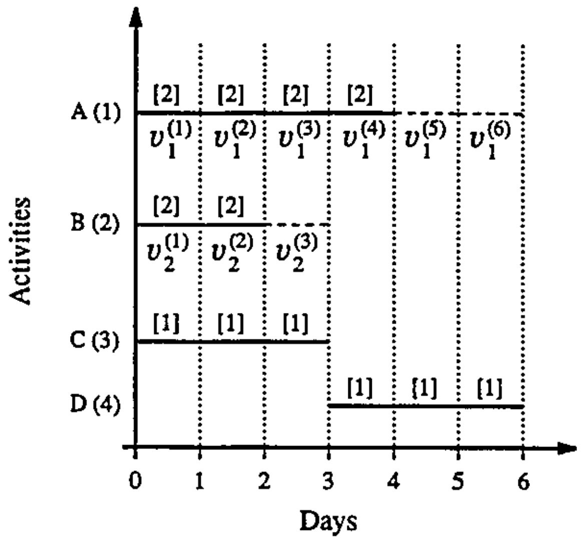
This means that, after leveling, the start time of activity A has been shifted to day 3 (i.e., $v_1^{(3)} = 1$) , while the start time of activity B has not been changed compared to ESS (i.e., $v_2^{(1)} = 1$). The convergence of the energy function, illustrated in Figure 7.4(b), demonstrates the fact that the model is quite robust, since it did not get stuck into the local minimum, and it finally reached the global minimum. Again the proposed ANN model has reached an optimal solution, which is illustrated in Figure 7.3(d).

7.4 Case Example 3

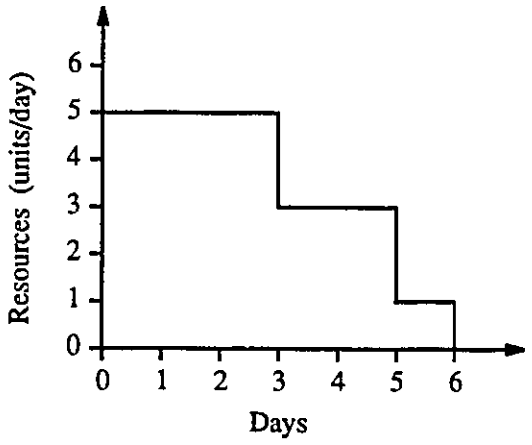
Example 7.3 The project-diagram of the third case example consists of five activities (J=5), A, B, C, D and E, and the critical path runs along activities D and E, as



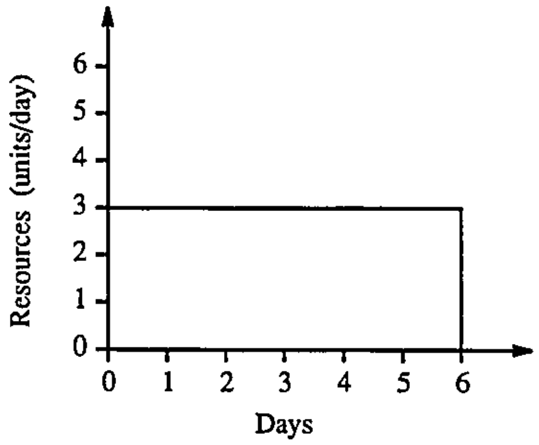
(a) Activity-on-arrow diagram



(b) Early start schedule

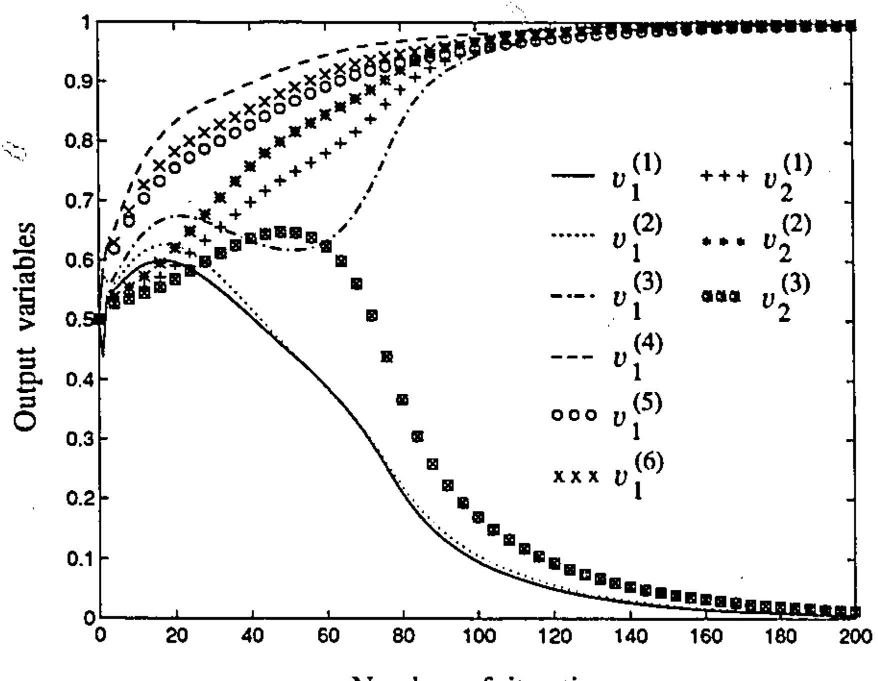


(c) Histogram of required resources before leveling

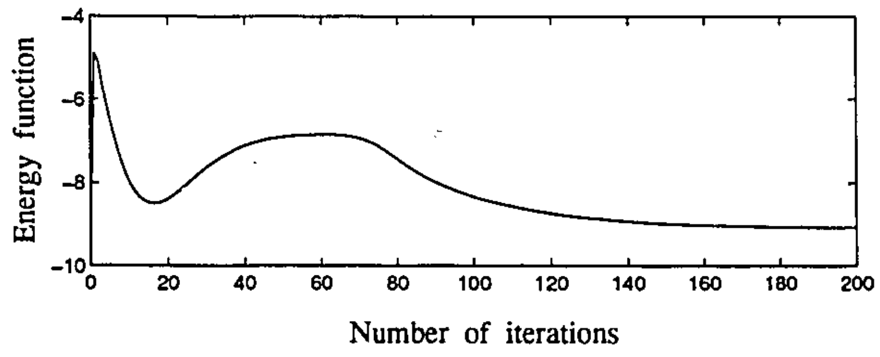


(d) Histogram of required resources after leveling

Figure 7.3: Resource leveling for Case Example 2



(a) Convergence of the output variables



(b) Convergence of the energy function

Figure 7.4: An illustration of the convergence of the proposed NN for Example 2

shown in Figure 7.5(a). The ESS is illustrated in Figure 7.5(b), and the histogram of required resources before leveling is shown in Figure 7.5(c). The project duration is K=5 days. In this example activities D and E are critical, while activities A, B, and C are noncritical, as illustrated in Figure 7.5(b). The start and finish times of activity A(1) are : $K_1(1) = 1$, and $K_2(1) = 4$. The total float of activity A is three days. The start and finish times of activity B(2) are: $K_1(2) = 3$, and $K_2(2) = 4$. The total float of activity B is one day. Finally, the start and finish times of activity C(3) are: $K_1(3) = 4$, and $K_2(3) = 5$. The total float of activity C is also one day. Corresponding to noncritical activities, the design vector, defined in (5.42), is equal to

$$\mathrm{v} = \left[\, v_1^{(1)}\; v_1^{(2)}\; v_1^{(3)}\; v_1^{(4)}\; v_2^{(3)}\; v_2^{(4)}\; v_3^{(4)}\; v_3^{(5)} \,\right]^{\mathrm{T}}.$$

where

$$\left[\, v_1^{(1)}\; v_1^{(2)}\; v_1^{(3)}\; v_1^{(4)} \,\right]$$

corresponds to activity A(1),

$$\left[\, v_2^{(3)}\; v_2^{(4)} \,\right]$$

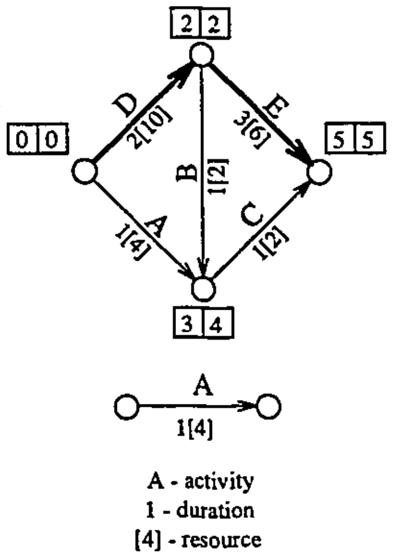
corresponds to activity B(2), and

$$\left[\, v_3^{(4)}\; v_3^{(5)} \,\right]$$

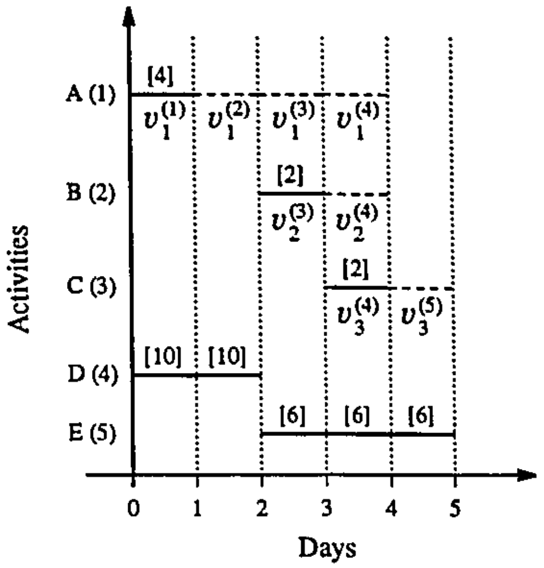
corresponds to activity C(3). As in the previous case examples, all the output variables have been initially assigned the arbitrary value of $\frac{1}{2}$. After leveling, their values have changed as illustrated in Figure 7.6(a).

$$\mathrm{v}_{\mathrm{final}} = \left[\, v_1^{(1)} = 0,\; v_1^{(2)} = 0,\; v_1^{(3)} = 1,\; v_1^{(4)} = 0,\; v_2^{(3)} = 0,\; v_2^{(4)} = 1,\; v_3^{(4)} = 0,\; v_3^{(5)} = 1 \,\right]^{\mathrm{T}}.$$

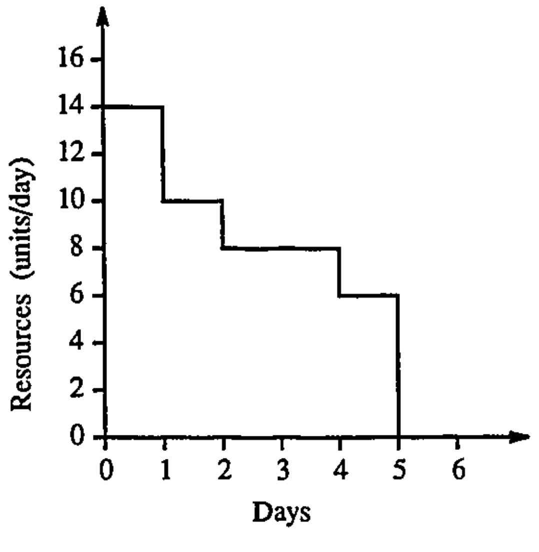
This means that, after leveling, the start time of activity A has been shifted to day 3 (i.e., $v_1^{(3)} = 1$), the start time of activity B has been shifted to day 4 (i.e., $v_2^{(4)} = 1$), and the start time of activity C has been shifted to day 5 (i.e., $v_3^{(5)} = 1$). The convergence of the energy function, illustrated in Figure 7.6(b), shows that the system required more time to reach an optimal solution than in the previous two cases, i.e.,



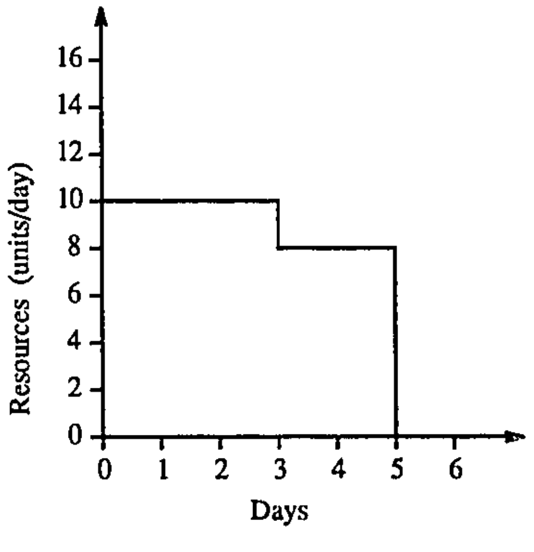
(a) Activity-on-arrow diagram



(b) Early start schedule

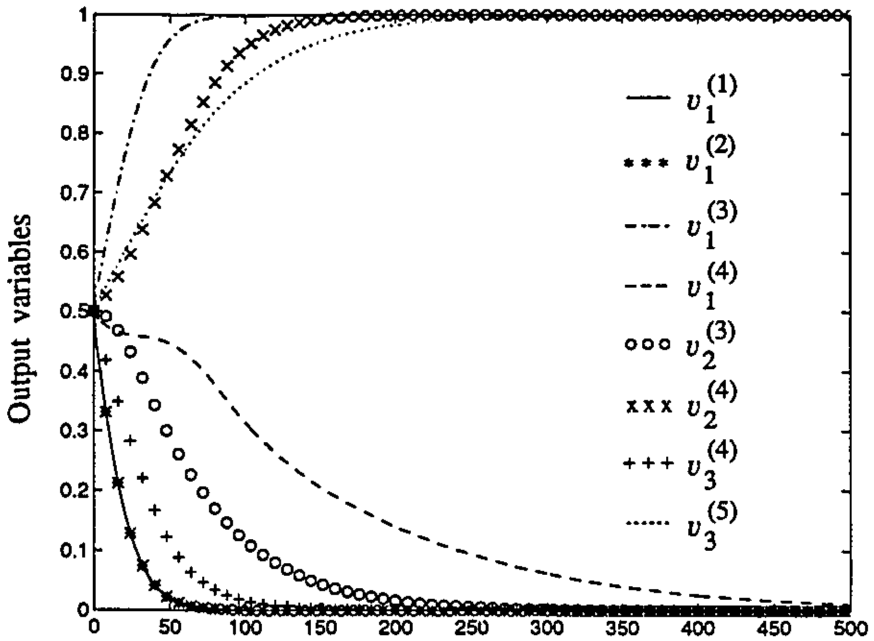


(c) Histogram of required resources before leveling

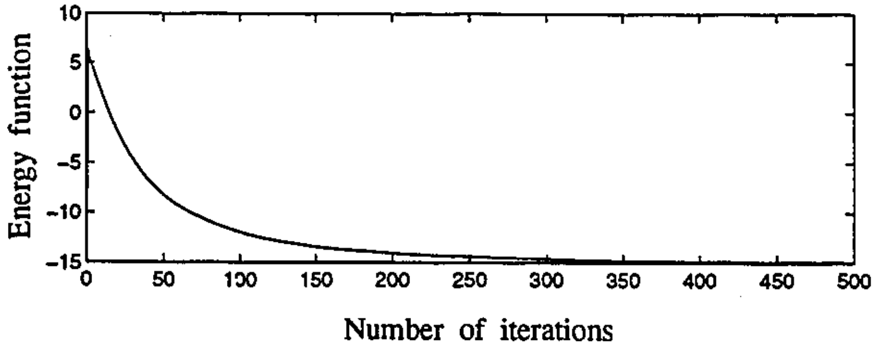


(d) Histogram of required resources after leveling

Figure 7.5: Resource leveling for Case Example 3



(a) Convergence of the output variables



(b) Convergence of the energy function

Figure 7.6: An illustration of the convergence of the proposed NN for Example 3

it needed almost 500 iterations to find its equilibrium point. In this example case, the proposed ANN model has reached an optimal solution, which is illustrated in Figure 7.5(d). As it is shown in the resource histogram after leveling, the required resources are utilized in a smoother fashion than before leveling, and the fluctuations in the pattern of resource usage have been reduced. This solution is found to be identical to the optimum solution for minimizing daily resource variations obtained by Easa (1989), which confirms the validity of the proposed ANN model.

Theoretically, the model can be used for any construction project, but practically it is constrained by the limits of the size of the matrices in Matlab, and by the choice of the Lagrange multipliers. Since there is no training involved, when the project changes, the ANN architecture remains unchanged, and only two steps need to be performed. The first one is straightforward: the user has to specify the new duration, the resource rate and the ESS values for each activity from previous CPM calculations, based on which the weight matrices are calculated. On the other hand, the second step is more complex: the user has to choose the initial values for the Lagrange multipliers, since these values vary for different sizes of the project-networks. As there is yet no methodology available as to what values the Lagrange multipliers ought to take (Gill, Murray and Wright 1993), several runs were made, for different sized-projects, varying the values of these multipliers. The model was found to be quite sensitive to the initial choice of the Lagrange multipliers, and it reached an optimal solution only when the multipliers were initialized with values as indicated in Section 4.3. When poor values were chosen, the equilibrium point was found to be far from an optimal solution.

The ANN formulation of the QALM optimization is a heuristic procedure for solving the RL problem. Therefore, in general, the proposed model will not necessarily find an optimal solution. Nevertheless, due to the computational speed which can be achieved using ANNs, the possibility of solving the RL problem using a neural network deserves further investigation.

Chapter 8

Concluding Remarks

8.1 Conclusions

The primary contributions of this work have been the development of a new neural network model aimed at solving the RL problem in construction , and the derivation of a new methodology for calculating the weight-matrix of a Hopfield NN for RL. The ANN model has been derived by mapping a formulation of the RL problem as a QALM optimization, onto an ANN architecture employing a Hopfield configuration of NN. It has been shown that the augmented Lagrangian associated to the RL problem can be interpreted as the energy function of the Hopfield NN. The ANN architecture consists of two main blocks: the Hopfield NN block, and a control block for the computation of the weights of the Hopfield network and for updating these weights according to the adjustment of Lagrange multipliers in the QALM optimization.

The new methodology for the derivation of the weight-matrices of the Hopfield NN has been developed based on the structural properties of the matrices arising from the formulation of the RL problem as a QALM optimization. These structural properties have been formalized as template-matrix contributions of different terms of the AL associated with the RL problem, to the entries of the weight-matrices. In the case of a specific project-diagram, several examples illustrating the structure of the template matrices have been provided. It has been emphasized that the template matrices can be filled-in easily, using the ESS of a project.

The theoretical framework described in this thesis allows for the derivation of a computationally-efficient yet versatile algorithm, which can easily accomodate different structures of project-networks, for the computation of the weight-matrix of a Hopfield-based NN architecture for resource leveling. With such an algorithm, the weight-matrix is filled-in directly by using characteristic templates, and without a need to explicitly determine the functional expressions of the weights.

An experimental validation of the proposed ANN model has also been carried out, by considering three case examples, and the experimental results have demonstrated the functionality of the proposed model. Further investigation is required, in order to verify the effectiveness of the proposed ANN model, in the case of medium and large-sized projects.

8.2 Scope for Future Study

The ANN model has only been preliminarily tested on small-sized projects. Further investigation is required in order to verify the effectiveness of the proposed ANN model in the case of medium and large-sized projects.

Also, a performance comparison of the proposed model versus heuristic search and genetic algorithm approaches is required, to better evaluate the proposed methodology.

In addition, the current application of the ANN model to AoA networks could be extended to PDM networks.

Furthermore, a further research can be undertaken for finding a parallel implementation of the model.

References

[1] ANSI/IEEE Standard 729-1983. *Standard Glossary of Software Engineering Terminology.* IEEE Press, 1983.

[2] J. J. Adrian. *Quantitative Methods in Construction Management.* American Elsevier Publishing Co., Inc., New York, 1973.

[3] H. N. Ahuja. *Construction performance control by networks.* John Wiley & Sons, New York, 1976.

[4] H. N. Ahuja, S. P. Dozzi, and S. M. Abourizk. *Project Management. Techniques in Planning and Controlling Construction Projects.* John Wiley & Sons, New York, second edition, 1994.

[5] S. I. G. Allam. Multi-project scheduling: a new categorization for heuristic scheduling rules in construction scheduling problems. *Construction Management and Economics*, 6:93–115, 1988.

[6] J. M. Antill and R. W. Woodhead. *Critical Path Methods in Construction Practice.* John Wiley & Sons, New York, fourth edition, 1990.

[7] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* Academic Press, New York, 1982.

[8] A. R. Burgess and J. B. Killebrew. Variation in activity level on a cyclical arrow diagram. *Journal of Industrial Engineering*, 13(2):76–83, 1962.

[9] D. G. Carmichael. *Construction Engineering Networks: Techniques, Planning and Management.* Ellis Horwood Limited, Chichester, England, 1989.

[10] T. C. Chang, C. W. Ibbs, and K. C. Crandall. Network resource allocation with support of a fuzzy expert system. *Journal of Construction Engineering and Management, ASCE*, 116(2):239–260, 1990.

[11] L. Chao and M.J. Skibniewski. Estimating construction productivity: neural-network-based approach. *Journal of Computing in Civil Engineering, ASCE*, 8(2):234–251, 1994.

[12] A. Cichocki and R. Unbehauen. *Neural networks for optimization and signal processing.* John Wiley & Sons, New York, 1993.

[13] E. W. Davis and J. H. Patterson. A comparison of heuristic and optimum solutions in resource constrained project scheduling. *Management Science*, 21:944–955, 1975.

[14] S. M. Easa. Resource leveling in construction by optimization. *Journal of Construction Engineering and Management, ASCE*, 115(2):302–316, 1989.

[15] D. Echeverry, C. W. Ibbs, and S. Kim. Construction schedule generation using AI tools. In *Proceedings of the Sixth Conference on Computing in Civil Engineering*, volume 1, pages 44–51, Atlanta, Georgia, September 1989. ASCE.

[16] Ch. Ephremidis. An integrated computer aided algorithm for optimal resource allocation. In B. V. H. Toppping, editor, *Proceedings of the Third International Conference on Civil and Structural Engineering*, volume 1, pages 71–74, London, England, August 1987. CIVIL-COMP Press.

[17] Q. W. Fleming, J. W. Bronn, and G. C. Humphreys. *Project and Production Scheduling.* Probus Publishing Co., Chicago, 1987.

[18] I. Flood. A neural network approach to the sequencing of construction tasks. In *Proceedings of the Sixth International Symp. on Automation and Robotics in Construction*, volume 1, pages 204–211, Austin, Texas, June 1989. ISARC.

[19] I. Flood and N. Kartam. Neural networks in civil engineering. II: Systems and application. *Journal of Computing in Civil Engineering, ASCE*, 8(2):150–163, 1994.

[20] J. A. Freeman and D. M. Skapura. *Neural Networks. Algorithms, Applications, and Programming Techniques.* Addison-Wesley, Reading, Mass., 1992.

[21] R. V. Galbreath. Computer program for leveling resource usage. *Journal of the Construction Division, ASCE*, 91(CO1):107–124, 1965.

[22] J. Garrett Jr. Neural networks and their applicability within civil engineering. In *Proceedings of the Eighth Conference on Computing in Civil Engineering*, volume 2, pages 1155–1162, Dallas, Texas, June 1992. ASCE.

[23] A. M. Geoffrion and R. E. Marsten. Integer programming algorithms: a framework and state-of-the-art survey. *Management Science*, 18(9):465–490, 1972.

[24] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, New York, tenth edition, 1993.

[25] S. Gulati, S. S. Iyengar, N. Toomarian, V. Protopescu, and J. Barhen. Nonlinear neural networks for deterministic scheduling. In *Proceedings of the First International Conference on Neural Networks*, volume 4, pages 745–752, San Diego, California, 1987. IEEE.

[26] F. Harris and R. McCaffer. *Modern Construction Management.* BSP Professional Books, Oxford, England, third edition, 1989.

[27] R. B. Harris. *Precedence and Arrow Networking Techniques for Construction.* John Wiley & Sons, New York, 1978.

[28] R. B. Harris. Packing method for resource leveling (PACK). *Journal of Construction Engineering and Management, ASCE*, 116(2):331–350, 1990.

[29] R. Hecht-Nielsen. Neurocomputing: Picking the human brain. *IEEE Spectrum*, 25(3):36–41, 1988.

[30] T. Hegazy, P. Fazio, and O. Moselhi. Developing practical neural network applications using backpropagation. *Micromputers in Civil Engineering*, 9:145–159, 1994.

[31] C. Hendrickson and T. Au. *Project Management for Construction. Fundamental Concepts for Owners, Engineers, Architects, and Builders.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[32] C. Hendrickson, C. Zozaya-Gorostiza, D. Rehak, E. Baracco-Miller, and P. Lim. Expert system for construction planning. *Journal of Computing in Civil Engineering, ASCE*, 1(4):253–269, 1987.

[33] J.J. Hopfield and D.W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[34] F. A. Karaa and A. Y. Nasr. Resource management in construction. *Journal of Construction Engineering and Management, ASCE*, 112(3):302–316, 1986.

[35] S. Karshenas and D. Haber. Economic optimization of construction project scheduling. *Construction Management and Economics*, 8(2):136–146, 1990.

[36] S. Karshenas and F. Xin. Application of neural networks in earthmoving equipment production estimating. In *Proceedings of the Eighth Conference in Computing in Civil Engineering and Geographic Information Systems*, volume 1, pages 841–847, New York, N.Y., 1992. ASCE.

[37] N. Kartam, I. Flood, and T. Tongthong. Artificial neural networks and knowledge-based systems: Complementary AI techniques for civil engineering.

In *Proceedings of the Fifth International Conference on Computing in Civil and Building Engineering*, volume 2, pages 1398–1403, Anaheim, California, June 1993. ASCE.

[38] T. C. Kavanagh, F. Muller, and J. J. O'Brien. *Construction Management. A professional approach.* McGraw-Hill, New York, 1978.

[39] Y. Kobayashi and H. Nonaka. Application of neural network to schedule integration in plant engineering. In *Proceedings of the International Neural Network Conference*, volume 1, pages 287–291, Paris, France, July 1990. INNS, Kluwer Academic Publishers.

[40] I. Kurtulus and E. W. Davis. Multiproject scheduling categorization of heuristic rule performance. *Management Science*, 28:161–172, 1982.

[41] R. C. Leachman. Multiple resource leveling in construction systems through variation of activity intensities. *Naval Research Logistics Quarterly*, 30:187–198, 1983.

[42] A. Lester. *Project Planning and Control.* Butterworth Scientific, London, England, 1982.

[43] R. E. Levitt and J. C. Kunz. Using knowledge of construction and project management for automated schedule updating. *Project Management Journal*, 16(5):57–76, 1985.

[44] F. K. Levy, G. L. Thompson, and J. D. Wiest. Multiship, multishop, workload-smoothing program. *Naval Research Logistics Quarterly*, 9(37):37–44, 1963.

[45] P. Lorterapong. *Fuzzy Project-Network Scheduling Under Resource Constraints.* PhD thesis, Concordia University, Montréal, Québec, Canada, July 1995.

[46] J. Martinez and P. Ioannou. Resource leveling based on the modified minimum moment heuristic. In *Proceedings of the Fifth International Conference*

on Computing in Civil and Building Engineering, volume 1, pages 287–294, Anaheim, California, June 1993. ASCE.

[47] M. J. Mawdesley and V. Carr. Artificial neural networks for construction project planning. In B. V. H. Toppping and A. I. Khan, editors, *Proceedings of the Third International Conference on The Application of Artificial Inteligence to Civil and Structural Engineering*, pages 39–46, Edinbourgh, U. K., August 1993. CIVIL-COMP Press.

[48] J. J. Moder, C. R. Phillips, and E. W. Davis. *Quantitative Methods in Construction Management*. Van Nostrand Reinhold Co., Inc., New York, third edition, 1983.

[49] A. A. Morad and M. C. Vorster. Network-based versus AI-based techniques in project planning. *Project Management Journal*, 24(1):23–30, 1993.

[50] O. Moselhi and K. El-Rayes. Scheduling of repetitive projects with cost optimization. *Journal of Construction Engineering and Management, ASCE*, 119(4):681–697, 1993.

[51] O. Moselhi, T. Hegazy, and P. Fazio. Neural networks as tools in construction. *Journal of Construction Engineering and Management, ASCE*, 117(4):606–625, 1991.

[52] O. Moselhi and P. Lorterapong. Least impact algorithm for resource allocation. *Canadian Journal of Civil Engineering*, 20(2):180–188, 1993.

[53] O. Moselhi and M. Nicholas. Hybrid expert system for construction planning and scheduling. *Journal of Construction Engineering and Management, ASCE*, 116(2):221–238, 1990.

[54] K. K. Movassaghi and S. A. Beidoun. A neural network model for construction resource leveling. In *Proceedings of the Fifth Conference on Computing in Civil Engineering*, volume 1, pages 381–390, Alexandria, VA, March 1988. ASCE.

[55] V. Navinchandra, D. Sriram, and R. D. Logcher. GHOST: Project network generator. *Journal of Computing in Civil Engineering, ASCE*, 2(3):239–254, 1988.

[56] J. O'Brien. VPM scheduling for high-rise buildings. *Journal of the Construction Division, ASCE*, 101(CO4):895–905, 1975.

[57] S. S. Panwalkar and W. Islander. A survey of scheduling rules. *Operations Research*, 25:45–61, 1977.

[58] J. H. Patterson. A comparison of exact approaches for solving the multiple constrained resource project scheduling problem. *Management Science*, 30(7):854–867, 1984.

[59] C. Peterson and J. R. Anderson. Neural networks and NP-complete optimization problems. *Complex Systems*, 2:59–89, 1991.

[60] R. Petrovic. Optimization of resource allocation in project planning. *Operations Research*, 16(3):559–568, 1968.

[61] D. T. Phillips and A. Garcia-Diaz. *Fundamentals of Network Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[62] R. N. Ramlogan and I. C. Goulter. Mixed integer model for resource allocation in project management. *Journal of Engineering Optimization*, 15:97–111, 1989.

[63] D. Savin, S. T. Alkass, and P. Fazio. A neural network model for construction resource leveling. In *Proceedings of the Annual Conference of the Canadian Society for Civil Engineering*, volume 1, pages 393–402, Winnipeg, Manitoba, June 1994. CSCE.

[64] D. Savin, S. T. Alkass, and P. Fazio. Formulation of the weight-matrix of a neural network for resource leveling. In B. V. H. Toppping, editor, *Proceedings of the Fourth International Conference on The Application of Artificial Inteligence to Civil and Structural Engineering*, pages 95–100, Cambridge, England, August 1995. CIVIL-COMP Press.

[65] J. E. Seibert and G. W. Evans. Time-constrained resource leveling. *Journal of Construction Engineering and Management, ASCE*, 117(3):503–520, 1991.

[66] K. A. Shah, F. Farid, and J. W. Baugh Jr. Optimal resource leveling using integer-linear programming. In *Proceedings of the Fifth International Conference on Computing in Civil and Building Engineering*, volume 1, pages 501–508, Anaheim, California, June 1993. ASCE.

[67] T. Shimazaki, K. Sano, and Y. Tuchiya. Resource leveling in PERT by neural networks. In G. Rzevski and R. A. Adey, editors, *Proceedings of the Sixth Conference on Applications of Artificial Intelligence in Engineering*, volume 1, pages 487–498, Oxford, U. K., June 1991. Elsevier Applied Science.

[68] F. B. Talbot and J. H. Patterson. Optimal methods for scheduling projects under resource constraints. *Project Management Quarterly*, pages 26–33, December 1979.

[69] D.W. Tank and J.J. Hopfield. Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, 33(5):533–541, 1986.

[70] A. Thesen. A manpower leveling/project planning program. *Journal of Computers and Industrial Engineering*, 2(3):115–122, 1978.

[71] V. Vemuri. *Artificial Neural Networks: theoretical concepts.* Computer Society Press of the IEEE, Washington, D.C., 1988.

[72] X. Wei and G. Singh. Neural networks for multiobjective and multiresource decision support system. In B. V. H. Toppping, editor, *Proceedings of the Fourth International Conference on The Application of Artificial Inteligence to Civil and Structural Engineering*, pages 127–135, Cambridge, England, August 1995. CIVIL-COMP Press.

[73] J. D. Wiest and F. K. Levy. *A Management Guide to PERT/CPM.* Prentice-Hall, Englewood Cliffs, NJ, 1977.

[74] G.V. Wilson and G.S. Pawley. On the stability of the travelling salesman problem algorithm of hopfield and tank. *Biological Cybernetics*, 58:63–70, 1988.

[75] B. M. Woodworth and C. J. Willie. A time-constrained approach to resource leveling in multi-project scheduling. *Project Management Quarterly*, 7(2):26–33, 1976.

[76] J. M. Zurada. *Introduction to Artificial Neural Systems.* West Publishing Co., New York, 1992.