INTELLIGENT ELEVATOR CONTROL BASED ON ADAPTIVE LEARNING AND OPTIMISATION

by

Edzard Adolf Biermann Jordaan

Thesis presented in partial fulfilment of the requirements for the degree of Masters of Science in Electrical and Electronic Engineering with Computer Science in the Faculty of Engineering at Stellenbosch University

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Dr. PJ Randewijk

December 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: ....2014/09/30..........

Abstract

INTELLIGENT ELEVATOR CONTROL BASED ON ADAPTIVE LEARNING AND OPTIMISATION

E.A.B. Jordaan

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: M.Eng

October 2014

Machine learning techniques have been around for a few decades now and are being established as a pre-dominant feature in most control applications. Elevators create a unique control application where traffic flow is controlled and directed according to certain control philosophies. Machine learning techniques can be implemented to predict and control every possible traffic flow scenario and deliver the best possible solution. Various techniques will be implemented in the elevator application in an attempt to establish a degree of artificial intelligence in the decision making process and to be able to have increased interaction with the passengers at all times.

The primary objective for this thesis is to investigate the potential of machine learning solutions and the relevancy of such technologies in elevator control applications. The aim is to establish how the research field of machine learning, specifically neural network science, can be successfully utilised with the goal of creating an artificial intelligent (AI) controller. The AI controller is to adapt to its existing state and change its control parameters as required without the intervention of the user.

The secondary objective for this thesis is to develop an elevator model that represents every aspect of the real-world application. The purpose of the model is to improve the accuracy of existing theoretical and simulated models, by modulating previously unknown and complex variables and constraints. The aim is to create a complete and fully functional testing platform for developing new elevator control philosophies and testing new elevator control mechanisms.

To achieve these objectives, the main focus is directed to how waiting time, probability theory and power consumption predictions can be optimally utilised by means of machine learning solutions. The theoretical background is provided for these concepts and how each subject can potentially influence the decision making process. The reason why this approach has been difficult to implement in the past, is possibly mainly due to the lack of adequate representation for these concepts in an online environment without the continuous feedback from an Expert System. As a result of this thesis, the respective online models for each of these concepts were successfully developed in order to deal with the identified shortcomings.

The developed online models for projected waiting times, probability networks and power consumption feedback were then combined to form a new Intelligent Elevator Controller (IEC) structure as opposed to the Expert System approach, mostly used in present computer based elevator controllers.

Uittreksel

INTELLIGENTE HYSBAKBEHEERDER GEBASEER OP AANGEPASTE LEER EN OPTIMALISASIE

E.A.B. Jordaan

Departement Elektriese en Elektroniese Ingenieurswese,
Universiteit Stellenbosch,
Matieland 7602, Suid-Afrika .

Tesis: M.Ing

Oktober 2014

Masjienleertegnieke bestaan al vir 'n paar dekades en is 'n oorwegende kenmerk in hedendaagse beheertoestelle. Hysbakke skep 'n unieke beheertoepassing, waar verkeersvloei beheer en gerig kan word volgens sekere beheerfilosofië. Masjienleertegnieke kan geïmplementeer word om elke moontlike verkeersvloei situasie te voorspel en te beheer en die beste moontlike oplossing te lewer. Verskeie tegnieke sal in die tesis ondersoek word in 'n poging om 'n mate van kunsmatige intelligensie in die besluitneming proses te skep asook verhoogte interaksie met die passasiers te alle tye.

Die primêre doel van hierdie tesis is om die potensiaal van 'n masjienleer oplossing en die toepaslikheid van dit in hysbakbeheertoepassings te ondersoek. Die doel is om vas te stel hoe die navorsing in die veld van die masjienleer, spesifiek in neurale netwerk wetenskappe, suksesvol aangewend kan word met die doel om 'n kunsmatige intelligente beheerder te skep. Die kunsmatige intelligente beheerder moet kan aanpas by sy onmidelike omgewing en sy beheer parameters moet kan verander soos nodig sonder die ingryping van die gebruiker.

Die sekondêre doelwit vir hierdie tesis is om 'n hysbakmodel, wat elke aspek van die werklike wêreld verteenwoordig, te ontwikkel. Die doel van die model is om die akkuraatheid van die bestaande teoretiese en gesimuleerde modelle te verbeter deur voorheen onbekende en komplekse veranderlikes en beperkings in ag te neem.

Die doel is om 'n funksionele toetsplatform te skep vir die ontwikkeling van nuwe hysbakbeheerfilosofië en vir die toets van nuwe hysbakbeheermeganismes.

Om hierdie doelwitte te bereik, is die hooffokus gerig om wagtyd, waarskynlikheidsteorie en kragverbruik voorspellings optimaal te gebruik deur middel van die masjienleer oplossings. Die teoretiese agtergrond is voorsien vir hierdie konsepte en hoe elke konsep potensieel die besluitneming kan beïnvloed. Die rede waarom hierdie benadering moeilik was om te implementeer tot hede, is moontlik te wyte aan die gebrek aan voldoende verteenwoordiging vir hierdie konsepte in 'n aanlynomgewing sonder die voortdurende terugvoer van 'n Deskundige Stelsel. As gevolg van hierdie tesis word die onderskeie aanlynmodelle vir elk van hierdie konsepte suksesvol ontwikkel om die geïdentifiseerde tekortkominge te oorkom.

Die ontwikkelde aanlynmodelle vir geprojekteerde wagtye, waarskynlikheidsnetwerke en kragverbruik terugvoer is dan gekombineer om 'n nuwe intelligente hysbakbeheerder struktuur te skep, in teenstelling met die Deskundige Stelsel benadering in die huidige rekenaar gebaseerde hysbakbeheerders.

Acknowledgements

All acknowledgements are given to my Heavenly Father, for granting me the ability and the favorable circumstances to complete this thesis. Sincere gratitude goes out to my family and friends for all their support and motivation, my Supervisor for his guidance and to Majuba personnel and PTM for their insight and assistance.

Contents

Declaration i

Abstract ii

Uittreksel iv

Acknowledgements vi

Contents vii

List of Figures x

List of Tables xiii

Listings xv

Nomenclature xvi

1 INTRODUCTION 1

1.1 INTRODUCTION 1
1.2 THESIS OBJECTIVES 1
1.3 THESIS CONTRIBUTIONS 2
1.4 THE RESEARCH ENVIRONMENT 3
1.5 THESIS OVERVIEW 3

2 LITERATURE REVIEW 5

2.1 ELEVATOR OPERATING SYSTEM AND CONTROL 5
2.2 ONLINE TRANSPORTATION PROBLEMS 9
2.3 MACHINE LEARNING ALGORITHMS AND TECHNIQUES . 11
2.4 SUMMARY AND CONCLUSIONS 24

3 ELEVATOR CONTROL: INTRODUCTION TO ARTIFICIAL INTELLIGENCE 25

3.1 INTRODUCTION TO ARTIFICIAL INTELLIGENCE 25
3.2 INTELLIGENT SINGLE ELEVATOR CONTROL: GENERAL PHILOSOPHY 26

3.3 ELEVATOR MODE PERSONALISATION 29
3.4 INTELLIGENT ELEVATOR GROUP CONTROL 30
3.5 VARIATIONS AND CONSTRAINTS 31
3.6 INFORMATION RECONCILIATION 33
3.7 SUMMARY AND CONCLUSIONS 33

4 ELEVATOR POWER CONSUMPTION 34
4.1 POWER CONSUMPTION PHILOSOPHY 34
4.2 ELEVATOR MODEL 35
4.3 MEASUREMENT AND VERIFICATION 39
4.4 MOTOR DESCRIPTION 46
4.5 SUMMARY AND CONCLUSIONS 55

5 ELEVATOR PROBABILITY THEORY 60
5.1 PROBABILITY PHILOSOPHY 60
5.2 FLOOR ORDER PROBABILITY 61
5.3 EXPECTED REQUESTS 64
5.4 CAR CAPACITY PROBABILITY 68
5.5 SUMMARY AND CONCLUSIONS 72

6 WAITING TIMES CONSTRAINTS 73
6.1 WAITING TIME PHILOSOPHY 73
6.2 BASIC WAITING TIME CALCULATIONS 74
6.3 WAITING TIME CONJOINING NETWORK 75
6.4 SUMMARY AND CONCLUSIONS 80

7 NEURAL NETWORK ELEVATOR INTEGRATION 82
7.1 INTRODUCTION 82
7.2 NEURAL NETWORK ENERGY MODEL 83
7.3 NEURAL NETWORK PASSENGER DEMAND AND TRAFFIC STATE CLASSIFICATION 92
7.4 CAR CAPACITY CLASSIFICATION 96
7.5 AUTOMATED REASONING WITH NEURAL NETWORKS . 98
7.6 INTELLIGENT ELEVATOR CONTROLLER SUMMARY AND CONCLUSIONS 102

8 ELEVATOR CONTROL SIMULATION 104
8.1 TRAFFIC GENERATION AND COLLECTION 104
8.2 SINGLE ELEVATOR CONTROL SIMULATION 108
8.3 ELEVATOR GROUP CONTROL SIMULATION - WITHOUT DESTINATION DISPATCHING 123
8.4 ELEVATOR GROUP CONTROL SIMULATION - WITH DESTINATION DISPATCHING 128
8.5 SUMMARY AND CONCLUSIONS 134

9 CONCLUSIONS AND RECOMMENDATIONS 138
9.1 SUMMARY 138
9.2 THESIS CONTRIBUTION AND ORIGINAL CONTENT . . . 139
9.3 CONCLUSIONS 140
9.4 RECOMMENDATIONS FOR FUTURE WORK 144

List of References 145

List of Figures

2.1 Software Structure for The Computer Control of An Elevator System In The Late 1970's [1]. 6
2.2 Competitive Ratio of Smartstart Online Strategy, with $\rho > 1$. . . . 10
2.3 Fuzzification Flow Diagram. 14
2.4 Genetic algorithms (GA) General Flow Diagram. 17
2.5 Schematic Illustration of Two Biological Neurons [2]. 18
2.6 McCulloch and Pitts Model of a Single Neuron [2]. 18
2.7 A Multilayer Perceptron Neural Network Having Two Layers of Weights [2]. 19
2.8 A Hopfield Neural Network with 12 Connections [3]. 20
2.9 A Typical Feedforward Neural Network [3]. 21
2.10 Temperature-time Ranges of Various Conventional and Advanced Annealing Techniques [4]. 22
2.11 Evolution of The Elastic Net Over Time [5]. 24

4.1 Theoretical Geared Elevator Model 36
4.2 Measurement and Verification Energy Metering Process [6]. 40
4.3 Data Sampling with the OMICRON CMC 256-Plus Using Analog Clamp-on's. 42
4.4 Data Sampling with the OMICRON CMC 256 Plus. 42
4.5 Actual Energy Measurements taken at Unit 1 Machine Room. . . . 44
4.6 AC/DC Clamp Adaptor Used for Current Measurements. 44
4.7 Measurement and Verification Energy Measurement Panel Layout [7]. 45
4.8 Schindler Aux Bay Elevator Drive Machine, Located at Majuba Power Station, Unit 1. 47
4.9 Two Speed Motor Current Sampling. 49
4.10 Aux Bay U1 Elevator Motor Terminal Schematic. 49
4.11 Complete Torque-Speed Curve of a 3-Phase Induction Motor [8]. . . 50
4.12 Two Speed Motor Winding Transitioning Period. 50
4.13 Single-Phase Equivalent Circuit For a Polyphase Induction Motor [9]. 52
4.14 Main Supply Current Sample for a 20m to 16m Elevator Cycle Period. 53

4.15 18 Pole and 4 Pole Supply Current (rms) vs. Distance Travelled and Load Percentages (0-100% of rated load per trip distance). . . . 56
4.16 Elevator Supply Current (rms) and Input Power vs. Distance Travelled and Load Percentages (0-100% of rated load per trip distance). 57
4.17 Elevator Running Time vs. Distance Travelled and Load Percentages (0-100% of rated load per trip distance). 58

5.1 Floor Order Probability Computations 63
5.2 Poisson Probability Functions. 65
5.3 Actual non-homogeneous Poisson Passenger Arrivals (adaption from [10]. 69
5.4 Up-peak Passenger Arrivals (Generated Actuals vs. Predicted) . . . 70

6.1 RBF Graphical Illustrations: Waiting Time vs. Distance from Elevator Car. 78
6.2 Exponential RBF Interpolant with Gaussian Radial Basis Function. 79

7.1 K-means Energy Sample Extraction Process. 85
7.2 Neural Network Supply Current and Power Consumption Pattern Recognition. 91
7.3 Non-homogeneous Poisson Passenger Arrival Rates, Actual vs. Predicted for a 5 Day Period 95
7.4 Smartstart Load Capacity Setting's Effect on Overall Waiting Time 97
7.5 Dynamic Smartstart Load Capacity Setting's Effect on Waiting Time 98
7.6 RBF Network Representing Building Passenger Demand for Different Time Periods. 100
7.7 Intelligent Elevator Control Structure 103

8.1 Passenger Demand Benchmark for a Typical Building [11]. 106
8.2 % Population vs. Period for a Typical Building [11]. 107
8.3 General Traveling Salesman Problem Simulation Trough Brute Force.110
8.4 Passenger Waiting Times for a Worst Case Scenario: Baseline vs. TSP BF vs. TSP GA. 118
8.5 Elevator Controller Service Performance (with Service Time) for a Worst Case Scenario: Baseline vs. TSP BF vs. TSP GA. 120
8.6 Elevator Controller Service Performance (with Service Count) for a Worst Case Scenario: Baseline vs. TSP BF vs. TSP GA. 121
8.7 Actual Trip counter for the lunch hour period (Aux Bay U1, 11 Aug 14) 122
8.8 Intelligent Elevator Controller Graphical User Interface (version 1.0): Medium Rise Building Configuration. 125
8.9 False VRP Combinations with Binary Representation (4 Landings to Visit). 126
8.10 Service Waiting Time Performance for an Up-peak Cycle (Building Population Capacity of a 500 People). 131

8.11 Service Waiting Time Performance for an Up-peak Cycle (Building Population Capacity of a 1000 People). 135
8.12 Service Passenger Count Performance for an Up-peak Cycle (Building Population Capacity of a 1000 People). 136

List of Tables

2.1 The History of Available Group Elevator Control Technologies [1]. . 7
2.2 TSP: Number of Calculated Steps. 11
2.3 NP Complete and in-P Problem Examples [12]. 12
2.4 2 Dimensional Weight Matrix to Represent Each Connection and its Associated Weights [3]. 19

3.1 User Personalised Elevator Control Modes. 30

4.1 Bill of Material of The Energy Measurement Panel [7]. 46
4.2 Unit 1 Schindler Aux Bay 2-Speed Motor Nameplate Values 46
4.3 Theoretical Nameplate Motor Calculations. 48
4.4 Unit 1 Aux Bay Elevator Actual Values and Measurements. 48
4.5 Current Sample Descriptions, Refer to Figure 4.14. 53
4.6 Actual Measurements for 8 Persons Going Down From 20m to 16m , Refer to Figure 4.14. 54
4.7 Actual Measurements for 1 Person Going Down From 20m to 16m , Refer to Figure 4.14. 54

6.1 Exponential RBF Test Model Results. 77

7.1 Output Normaliser Error Performance: Neural Network Output vs. Training Data. 87
7.2 Input Normalisation: Training Data Pre-processing. 89

8.1 The Traveling Salesman GA Class Initialisation. 111
8.2 Algorithmic Performance Results for Single Elevator Control. . . . 116
8.3 Passenger Average Waiting Time Results for a Worst Case Scenario. 117
8.4 Elevator Controller Service Performance Results. 119
8.5 Energy Model Simulation Results for a 5 Day Period with Aux Bay U1 Elevator Configuration. 123
8.6 BF VRP: Number of Calculated Steps for a 5 Car Configuration. . 126
8.7 Elevator Controller Service Performance Results for a Group Elevator Configuration without Destination Dispatching and for a Building Population of 500. 132

8.8 Elevator Controller Service Performance Results for a Group Elevator Configuration with Destination Dispatching and for a Building Population of 500. 133

Listings

6.1 Gaussian Radial Base Function from Eq. 6.3.5 77
6.2 Interpolated Waiting Times. 78
8.1 Complete Randomisation Function. 104
8.2 Passenger Information Class Definitions. 107
8.3 Routing Time Estimation for Unit 2 Aux Bay. 108
8.4 Chromosome Decleration. 111
8.5 Chromosome Fitness Calculation. 112
8.6 Proximity Function. 112
8.7 Chromosome Mating Call Method. 112
8.8 TSP Simulated Annealing Parameters. 114
8.9 True VRP Combination Generator. 126

Nomenclature

Abbreviations and Acronyms

AC	Alternating Current
AI	Artificial Intelligence
ANN	Artificial Neural Network
BMS	Building Management System
CBR	Case Base Reasoning
CT	Current Transformer
DD	Destination Dispatching
DSP	Digital Signal Processing
DSRD	Dynamic Service Request Database
ED	Effective Duty
GA	Genetic Algorithms
GARS	Genetic Algorithm for Regressors' Selection
GARST	Genetic Algorithm for Regressors' Selection And Transformation
GUI	Graphical User Interface
HP	Horse Power
IAC	Integrated Access Control
IC	Integrated Chip
ICU	Intensive Care Unit
IEC	Intelligent Elevator Controller
IEC	International Electrotechnical Commission
ILP	Integer Linear Programming
IP	Ingress Protection
IP	Integer Programming
LIFO	Last In First Out
M&V	Measurement and Verification
MST	Minimum Spanning Tree
NARH	The Neural Automated Reasoning Head

NP	Non-deterministic Polynomial-time
NTDN	The Neural Traffic Distribution Network
OEM	Original Equipment Manufacturer
P	Polynomial-Time
PORT	Personal Occupant Requirement Terminal
PTSP	Probabilistic Traveling Salesman Problem
PTSPD	Probabilistic Traveling Salesman Problem with Deadlines
RBF	Radial Basis Function
RFID	Radio-frequency Identification
RMS	Root Mean Square
SAT	Satisfiability
SDVRP	As Site-Dependent Vehicle Routing Problem
TSP	Traveling Salesman Problem
VIP	Very Important Person
VRP	Vehicle Routing Problem
VT	Voltage Transformer
VVVF	Variable Speed Variable Frequency
WT	Waiting Time

Constants

g	9.81 m.s^{-2}

Symbols and Variables

α	constant parameters	
α	rotational acceleration	
β	constant parameters	
Δt_{moving}	time spend in the car while moving	[sec]
$\Delta t_{\text{stationary}}$	time spend in the car when stationary	[sec]
η	efficiency at rated speed	[%]
λ_i	arriving rate	
ω_{slip}	slip speed	[rad.s^{-1}]
ω_{sync}	stator angular speed	[rad.s^{-1}]
ω_m	rotor angular speed	[rad.s^{-1}]
$\overline{X_s}$	average	
$\phi_q(s)$	base function	
$\phi_q(x)$	localised function	

ρ	approximation ratio	
$\sigma_{x,s}$	standard deviation	
θ	scaling parameter	
a_{ij}	amount of available resources	
b_i	budget or the allowed time limit	[sec]
c	competitive ratio	
c_j	cost	
C_M	suspension ratio	
D_{ij}	passenger's final destination	
ds	sheave diameter	[cm]
E_{fullD}	energy for the full sample duration	[J]
$f(x)$	Poisson probability density	
$F(x)$	Poisson probability distribution	
I_{fullD}	current for the full sample duration	[A_{rms}]
I_{load}	load inertia	[kg.m^2]
I_A	phase current	[A]
I_N	rated current	[A]
K	the scale parameter	
$K_{i\_up}(x)$	passengers will not travel higher than the i^{th} floor	
K_i	population that have not reached floor i	
$K_i(x)$	passengers amount	
L_{cable}	lenght	[m]
l_{rope}	total rope length	[m]
m_{car}	no load car weight	[kg]
m_{cw}	counterweight mass	[kg]
m_{rated}	rated full load car weight	[kg]
m_{rope}	rope mass	[kg]
m_{car}	car mass	[kg]
m_{cw}	counterweight mass	[kg]
n	overall system efficiency	
N_{i_up}	total number of floors above the current floor	
n_1	friction pulley performance rate	
n_2	friction pulley benches performance rate	
n_3	worm screw performance rate	
n_f	forward efficiency	
n_r	feedback or reverse efficiency	

O_i	total population at floor i	
$P(F_{X,Y}\|F_Y)$	posteriori probabilities	
$P(F_Y\|F_{X,Y}$	transition probabilities	
$P(X)$	probability	
P_{fullD}	power for the full sample duration	[W]
P_{in}	active 3 phase input power	[W]
Q_{in}	reactive 3 phase power	[VAR]
r	radial separation	
r_0	scaling factor	
r_g	gearbox reduction ratio	
S_{in}	apparent 3 phase input power	[VA]
s_n	slip	[rad.s^{-1}]
T_{fullD}	time for the full sample duration	[sec]
t_{initial}	initial time recorded for a pre-set floor order . . .	[sec]
T_{rope}	actual breaking load of rope	[N]
T_e	torque at rated speed	[Nm]
U_i	number of people that are currently on floor i	
$V_{\text{Drop Value}}$	voltage drop constant per cable size	[mV/A/m]
V_{drop}	voltage drop	[V]
V_{ik}	value of of the k^{th} feature of case i	
V_C	contract speed	[m.s^{-1}]
V_m	rated motor speed	[rpm]
V_N	rated car speed	[m.s^{-1}]
w_{ij}	weight of city i to the point j.	
w_{kq}	weighing parameters	
X	data point	
$X_{i,\sigma}$	provides a Z-score	
x_j	passenger count	
Y_k	continuous differentiable surface	

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

Machine learning techniques have been around for a few decades now and are being established as a pre-dominant feature in most control applications. Elevators create a unique control application where traffic flow is controlled and directed according to certain control philosophies. Machine learning techniques can be implemented to predict and control every possible traffic flow scenario and deliver the best possible solution. Various techniques will be implemented in the elevator application in an attempt to establish a degree of artificial intelligence in the decision making process and to be able to have increased interaction with the passengers at all times.

1.2 THESIS OBJECTIVES

1.2.1 INTELLIGENT ELEVATOR CONTROL ASSESSMENT AND COMPARISON

The primary objective for this thesis is to investigate the potential of a machine learning solution and the relevancy of such technology in an elevator control application. The aim is to establish how the research field of machine learning and neural network science can be successfully utilised with the goal of creating an artificial intelligent controller. The AI controller is to adapt to its existing state and change its control parameters as required without the intervention of the user. The elevator model is to replicate exact real-time energy consumption values and waiting times, which are used to influence the decision making process of the controller in order to be a more energy efficient and optimal machine. The developed machine learning mechanisms should also be compared to existing control philosophies to establish competitiveness against a baseline.

1.2.2 VIRTUAL MODEL AND TESTING PLATFORM

The secondary objective is to have an elevator model that represents every aspect of the real-world application as accurately as possible. The model should improve the accuracy of existing theoretical or simulated models, by modulating previously unknown and complex variables and constraints. The model should include considerations to the elevator motor, drive, rope configuration and load variances. In addition to the required elevator model, a virtual environment should also be created; where multiple elevators are installed with the respective control mechanisms and integrated building management system (BMS). The virtual environment for elevator groups should include relevant building dynamics together with simulated population distributions. The aim is to create a complete and fully functional testing platform for developing new elevator control philosophies and testing new elevator technologies.

1.3 THESIS CONTRIBUTIONS

1.3.1 BENCHMARKING

The elevator's machine learning control can be compared and benchmarked against other philosophies and control technologies by way of simulation. Also by conducting energy consumption and waiting time analysis on the different elevator control techniques, it can illustrate the benefits and the need to upgrade the elevator's control system entirely.

1.3.2 GLOBAL ELEVATOR RESEARCH CONTRIBUTIONS

Information and results produced by this thesis can be a major contribution towards the research field of elevators. There are largely over 8.5 million elevators in operation worldwide, with the amount of new installations reaching 100 000 annually [13]. Each existing elevator also require an upgrade or a replacement every 20 to 25 years, because of equipment obsolescence or the possibility of appended statuary requirements. This creates a stable and continuous environment for elevator technology improvements and implementations thereof. It is thus important to produce a number of formal theses and registered articles to keep the research field updated and to keep it from stagnating.

1.4 THE RESEARCH ENVIRONMENT

The research environment for this thesis is directly applicable to Eskom Generation Power Stations, but not limited to any specific elevator configuration. The measurements and verification results for this thesis shall be specifically captured from Majuba Power Station, where 25 elevators are in operation.

1.5 THESIS OVERVIEW

The structure of the thesis is as follows:

Chapter 1 - INTRODUCTION
This chapter presents the thesis objectives, namely to develop an intelligent elevator controller based on machine learning techniques and to create a virtual model and testing platform. The thesis contributions are discussed and the research environment is defined

Chapter 2 - LITERATURE REVIEW
This chapter presents a brief introduction to elevator control possibilities and some background information on the history of elevator technologies. Various machine learning concepts and artificial intelligence philosophies are introduced in this chapter with the intent of implementing some of them in an elevator application. Mention is also made to online vs. offline strategies and the implementation thereof.

Chapter 3 - ELEVATOR CONTROL: INTRODUCTION TO AI
In this chapter, specific focus is placed on employing different machine learning techniques in order to improve the computational and symbolic intelligence of the elevator control system. The main focus is to obtain the best possible decisions with the available resources and limitations, but also to create an AI system that can provide results based on its own artificial intellect and not necessarily with pre-programmed responses. Specific references are made to the Traveling Salesman problem and the Vehicle routing problem that will be implemented throughout this thesis.

Chapter 4 - ELEVATOR POWER CONSUMPTION
This chapter will create an energy model for our elevator configuration based on theoretical calculations and actual energy samples.This chapter shall only focus on traction elevators, where a geared traction elevator is used for actual measurements, simulated results and theoretical models. The actual energy samples will be compared to the theoretical calculations based on the rated nameplate values, as well as the developed equations from this chapter.

Chapter 5 - ELEVATOR PROBABILITY THEORY
This chapter will focus on the probability philosophy definition and make various theoretical predictions related to the elevator application. The Poisson arrival probability function will also be introduced and implemented in such a way to be used as a building population generator and to be incorporated with other passenger probability definitions.

Chapter 6 - WAITING TIMES CONSTRAINTS
In this chapter the general waiting time philosophy will be provided, which states that the controller should be able to minimise overall waiting time of all passengers by establishing the most optimal route to follow for each car. The concept of time management will also introduced, instead of just looking at time minimisation techniques. A waiting time conjoining network will be developed in this chapter, by implementing radial base function (RBF) techniques.

Chapter 7 - NEURAL NETWORK ELEVATOR INTEGRATION
This chapter will describe in detail how effective Neural Networks can be in an elevator application and how Neural Networks will be at the core of an intelligent self-reasoning unit and provides how its predictive abilities can be used in our application.

Chapter 8 - ELEVATOR CONTROL SIMULATION
This chapter is to develop the testing platform and the elevator simulation with Java code, to be able to asses and compare various control philosophies and algorithms.

Chapter 9 - CONCLUSIONS AND RECOMMENDATIONS
The thesis will conclude with contributions made to the elevator research field and with any recommendation for future work.

Chapter 2

LITERATURE REVIEW

The purpose of this chapter is to provide a literature review on general machine learning techniques, with the purpose of familiarising the reader with the respective research fields which are relevant to this thesis and expose the reader to the various elevator control techniques being implemented at present.

2.1 ELEVATOR OPERATING SYSTEM AND CONTROL

2.1.1 ELEVATOR CONTROLLER

When the conventional control system was first introduced there were only a few distinct technologies available on the market that were being utilised by the various elevator manufacturing companies. Relay logic and later on solid state components were available for the implementation of an elevator control system. Solid state components employing integrated circuits provided higher reliability at the time, but with higher maintenance requirements due to increased complexity and a higher degree of knowledge that were required [1]. The major drawback with these two types of controller components are the lack of adaptability or flexibility to design changes. For each newly installed elevator system we would like to properly define the various group control mechanisms, like sectoring and up-peak with down-peak sub zoning etc. but with these technologies most settings are final after manufacturing stage and are difficult to fine-tune later. However solid state components provides a greater degree of flexibility than relay logic [1]. Other drawbacks to relay and solid state fixed logic are their inability to simulate new control algorithms and to prove their practicality without actual installation of the system. Programming and simulations were very much restricted in those days and elevator companies were limited to actual installations and costly modifications to evaluate and improve performance.

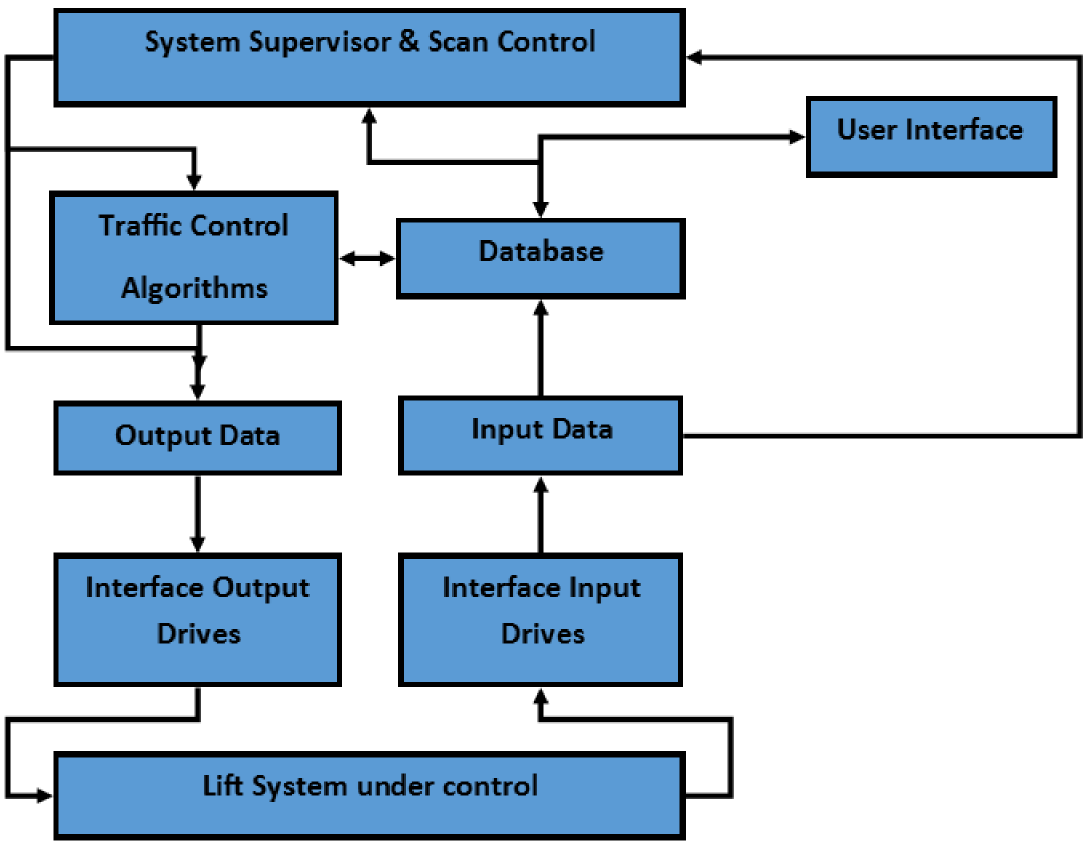


Figure 2.1: Software Structure for The Computer Control of An Elevator System In The Late 1970's [1].

Relay and solid state fixed logic technologies have since been replaced with ever improving digital computer control systems which have increased computational and speed abilities. Computer control allows for numerous simulations to be conducted for evaluation purposes and to test new algorithms and philosophies. Software control also enables on-line alterations to most parameters of the control algorithms throughout the design and operating cycle as required. Data logging is another advantage of computer control. Various traffic data can be logged, together with the elevator system responses and possible fault reporting integration.

2.1.2 ELEVATOR SYSTEM CONTROL

The basic single elevator car philosophy, generally found in old elevator installations can be described as follows: When the elevator car travels in the upwards direction, it stops at all the landing floors where the up directional button was pressed and vice versa. It will then change direction and service any other landing floors where a directional button was pressed. When the destination floor button is pressed inside the elevator car, that floor is visited if it is in conjunction with the direction being travelled. Similar single elevator car philosophies include: highest-floor-first and longest-waiting-passenger-first. Highest-floor-first algorithm will serve the highest activated floor first and travels downwards to the rest of the floor calls. The longest-waiting-passenger-first algorithm don't actually know how long the passenger is waiting, but rather the time difference between the first call was registered on the floor and the last time the floor was serviced.

The history of elevator group control technologies can be summarised in Table 2.1 [1]. The previous generations are not discussed here in detail, because we are rather interested in the current generation; which is computer aided elevator control.

Table 2.1: The History of Available Group Elevator Control Technologies [1].

Generation	Years	Control category
1	1850 - 1890	Simple mechanical control
2	1890 - 1920	Attendant and electrical car switch control
3	1920 - 1950	Attendant and push button control
4	1950 - 1975	Group control: scheduled and zoned
5	1975 - present	Computer group control

Group control is responsible to improve overall elevator system performance by having individual elevator cars working together to handle various traffic flow patterns and intensities. Different philosophies have been introduced in order to find the best joint solution for two or more elevator cars operating in the same conditions and passenger distributions.One of the more general philosophies is static-zoning, where common landings are grouped together based on physical proximities. Each elevator car only responds to the requests that originate from its designated cluster of floors and then transports all passengers in the elevator car to the respective destination floors. The requests can also be divided between up and down directions, where the elevator cars are divided into these two groups for responding to directional calls rather than actual floor positions. The landing floors can also be divided dynamically depending on the various elevator car positions. The idea is to have the elevator cars uniformly distributed and located throughout the length of the elevator's configuration. Each elevator car then only responds to the requests that are between the elevator car and the next car traveling in the same direction. When the top and bottom floors are reached the direction of the car interchanges. Any idle elevator car can either stay at its last destination floor with doors open or move to a designated floor, which is normally the main landing.

Most philosophies are designed with a response setting to different traffic patterns, to optimise control in high passenger demand states. Thus elevator control is normally defined in terms of different traffic modes. Up-peak mode can be defined as an elevator system state whereby high intensity up-peak traffic is imminent or already being experienced [1]. Up-peak traffic is where heavily or fully loaded cars are leaving the main landing at specific times in a day; where the passenger demand is at its highest. During this state all available elevator cars are directed to the main landing to provide additional capacity. Only requests originating from the main landing and inside the elevator cars are being responded to during this mode. The other requests are being ignored for the duration of the up-peak state. The available elevator cars can also be parked at the main landing until, for instance the 80 % capacity threshold is reached and then only will the doors close to service the passengers, in order to minimise traveling costs and reduce return trips. Up-peak mode is detected with either weighing devices or trip counters to establish that the system is experiencing heavy traffic from the main landing. The controller should also be able to detect when the traffic has subsided and that the up-peak mode can be cancelled and the normal operating mode reinstated.

Down-peak mode is defined in much the same way as the up-peak mode, only whereby the system is experiencing high intensity down-peak traffic. During this mode the elevator cars tend to reach full capacity from higher floors and are unable to handle any requests from floors lower down the building. In this state the system normally ignores requests from other floors requiring to travel upwards and also service the floors in a round-robin fashion in order to allow each floor a systematic opportunity to be serviced [1].

The drawback of some directional button philosophies discussed above is that the elevator car does not know when it is close to reaching its full capacity and thus will still continue to stop at each registered landing floor. Some systems do however utilise a weighing device that signals the controller to ignore any further requests when full capacity has been reached, excluding the requests that originated from inside the elevator car. These philosophies also do not register any valuable information about their passengers or the respective destinations before the passenger gets into the elevator car. However, in 1992 Schindler introduced the first practical destination dispatch system to the market: The Miconic 10 [14]. Schindler effectively introduced a new traffic control philosophy where the landing floor directional buttons were replaced with destination floor buttons instead. The breakthrough was established by the work of Dr. P. Friendli, who realised that the control system can benefit from a more advanced human interface system [14]. This alteration provides the controller with additional data and creates new traffic control possibilities. However, this technology was only designed for elevator groups where you have more than one elevator per landing floor and was used to effectively dispatch the designated elevators to certain destination floors to effectively reduce the passenger waiting times. At this stage the controller didn't utilise all the available information about its passengers and was only further developed and released with the second generation destination dispatch system: the Schindler ID in 2002 [14]. The system added the functionality of RFID technology, where passenger information is readily accessible and used for more effective dispatching and personalisation.

13 years after the Miconic 10 Schindler released their third generation destination dispatch system; the PORT (personal occupant requirement terminal) technology, which is defined as a Transit Management plan for a building. After a destination floor is entered into a 7" touch landing floor panel; the optimal and designated elevator number to travel with is displayed. The PORT system also employs RFID and proximity sensor technology and makes provision for disabled passengers through audible communication and easy-to-reach buttons [15]. The drawback to this philosophy is that the dynamic data capturing is not always dependable and is ever changing. In real world situations people do change their minds on the destination floor button pressed and also leave the landing queue prematurely. The destination floor is also pressed more than once in a futile attempt to make the elevator react faster to the call. Schindler does attempt to overcome this difficulties by using a weighing device inside the elevator car to validate and logically balance out the number of people entering and exiting on each floor against the number of times the landing buttons were pressed.

2.2 ONLINE TRANSPORTATION PROBLEMS

Online-dial-a-ride problems are defined as on-line transportation requests, specifying the objects that are to be transported with each respective destination and source information. Requests are received dynamically throughout the duration of execution and is not initially known to the server. The server or the controller has a few deterministic strategies on how to respond to the new on-line requests. The first strategy is to ignore any new request and complete the existing schedule or sequence as initially planned. After completion, the additional requests are then processed into a next sequence to execute. The second strategy is to recompute the existing sequence at every new request. This Replan strategy will re-optimise the sequence for all known requests and thus keep it up to date and optimal for the rest of the instance. The third strategy is a combination between the previous two and has been developed by Krumke S. which is called the Smartstart strategy [16].

The Smartstart strategy provides the server with a decision function that will either allow new requests to be included into the current schedule or not. The server also has the ability to wait a calculated period before starting with a computed schedule. The competitiveness of these strategies can be compared to the optimal off-line sequence when all requests are known beforehand. The Replan strategy has been proven by Krumke to have a competitive ratio of at most 3.5 times the optimal sequence and the Ignore strategy a competitive ratio of at most 2.5 [16].

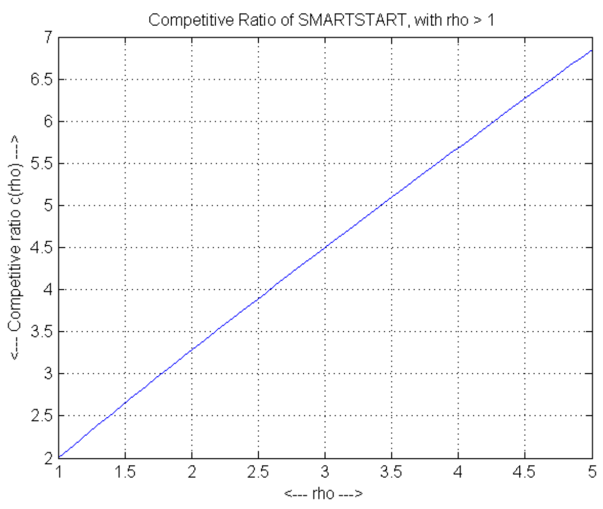


Figure 2.2: Competitive Ratio of Smartstart Online Strategy, with $\rho > 1$.

The Smartstart strategy can be proven to be the best possible online algorithm with a competitive ratio of [16],

$$c = \max\{\theta, \rho(1 + \frac{1}{\theta - 1}), \frac{\theta}{2} + \rho\} \ , \tag{2.2.1}$$

with ρ the ratio of the length as calculated from an approximation algorithm times the optimal length as calculated with brute-force. If the calculated length is in fact the optimal length, then $\rho = 1$. θ is the waiting scaling parameter and is defined as,

$$\theta > 1 \text{ with } t + l(S) \leq \theta t \ , \tag{2.2.2}$$

where t is the present time and $l(S)$ is the shortest schedule with the time difference between completion and start time t.

The best choice for θ is $0.5(1 + \sqrt{1 + 8\rho})$ which yields a competitive ratio of,

$$c(\rho) = 0.25(4\rho + 1 + \sqrt{1 + 8\rho}) \ , \tag{2.2.3}$$

with $\rho = 1$ and $\theta = 2$ the result will be a competitive ratio of 2, see Figure: 2.2 [16].

2.3 MACHINE LEARNING ALGORITHMS AND TECHNIQUES

2.3.1 THE TRAVELING SALESMAN PROBLEM

The Traveling Salesman Problem (TSP) is an "NP-hard (Non-deterministic Polynomial-time hard) problem which is an optimisation problem of finding the least-cost cyclic route through all nodes of a weighted graph" [17]. The problem can be generalised as follows: The traveling salesman must visit a number of cities, where he must identify the shortest overall route from one city to another where each city can only be visited once. The number of steps in the allocation process is factorial. If the number of cities is classified by n, the calculated steps will be $n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$ if calculated by brute force, see Table 2.2 [3].

The Vehicle Routing Problem (VRP) is an extension of the Traveling Salesman Problem, where there are a number of packages to be moved from a central depot to the destination location with a fleet of vehicles. Several variations or additions to this problem exist including, pickup and delivery, time windows, multiple trips and last in first out (LIFO). The aim is still to minimise the total route cost. Potvin and Thangiah [18] used genetic clustering effectively to first group certain nodes together based on each cluster's capacity limitations and pre-set criteria; for example, the nodes that have close proximity coordinates towards each other are grouped together. The number of clusters are set to the number of available vehicles. The shortest route is then calculated within each cluster, where it becomes the standard TSP again for each cluster. The implementation of the TSP can be extended to applications including computer wiring, machine sequencing and scheduling as well as frequency assignments in communication networks and statistical data analysis [19].

Table 2.2: TSP: Number of Calculated Steps.

Number of cities	Number of steps
1	1
2	1
3	6
4	24
5	120
6	720
7	5040
...	...
13	6,227,020,800

2.3.2 SEARCH AND OPTIMISATION PROBLEM SOLVING THEORIES

The Traveling Salesman Problem belongs to a group of search problems and concepts that have been around for decades and various attempts to solve these problems exist today, but neither of the attempted solutions have reached the satisfaction of our leading mathematicians and scientists of our time. A search problem is defined as an algorithm that is given an instance and a proposed solution as input and runs in polynomial time [12]. Table 2.3 provides a few of these problems grouped together as NP-complete and in-P problems. The right side refer to relatively easy problems which can be solved in polynomial (in-P) time and with dynamic and specialised algorithms [12]. On the left side we have problems which require much effort to solve effectively and often have running times of 2^n or worse. They can also be classified as decision problems or nondeterministic algorithms which produce accurate estimations or attempts to reduce the search spaces [12].

Mathematically, the problems from Table 2.3 are all related and can be reduced to or extended to suit the same mathematical model, thus it can prove to be beneficial to investigate them in an attempt to solve or improve our solution to the TSP. Thus each problem's terminology can be translated to the TSP structure which includes bridges, cities, paths, etc. SAT or satisfiability problems are Boolean statements or a collection of clauses which needs to be solved with true and false values. The 1-SAT problem (at most 1 positive literal), called the Horn formula, can be solved by a greedy algorithm [12]. 2-SAT refers to where you have two literals and a connection between them and can be effectively defined by image theory, however, 3 literals become more difficult (3-SAT) (refer to problem 1 and 10 in Table 2.3) [12].

Table 2.3: NP Complete and in-P Problem Examples [12].

No.	Problem	No.	Problem
1:	3SAT	10:	SAT, Horn SAT
2:	The traveling salesman problem	11:	Minimum spanning tree (MST)
3:	Longest path	12:	Shortest path
4:	3D matching	13:	Bipartite matching
5:	Knapsack	14:	Unary knapsack
6:	Independent set	15:	Independent set on trees
7:	Integer linear programming	16:	Linear programming
8:	Rudrata path/cycle	17:	Euler path
9:	Balanced cut	18:	Minimum cut

Related to the TSP is a much simpler problem, namely the minimum spanning tree (MST)(problem 11), with distance matrix and a bound b, where the total weight is defined as $\sum_{i,j} d_{ij} \leq b$. The difference is that the TSP is not allowed to branch like a tree, but rather be defined as a set path [12]. The Euler path (problem 17) refers to the problem where you have multiple edges between vertices or cities, where you have to find a path by crossing each edge (bridge) once, but you can visit each city more than once if required. The Rudrata cycle or path (problem 8) is similar to the Euler path, but only that every city must be visited only once with multiple bridge crossings allowed if required. A minimum cut problem (problem 19) is defined as the removal of a minimal number of edges to leave a graph disconnected, thus to isolate one or more cities from the rest of the cities. Balanced cut (problem 9) is where you need to partition or group different cities equally by cutting the bridges connecting the different groups. The vehicle routing problem can also be derived from the balanced cut problem theory. The Knapsack problem states that a hiker must decide how many goods to include in the trip vs. the comfort of carrying the goods. The hiker have a budget of b amount of weight allowed on the trip and more than one of each item is allowed, thus instead of $x = 0$ or $x = 1$ we have $x \geq 0$. (problem 5 and 14).

The TSP can also be defined by integer and linear programming techniques. The integer programming (IP) (problem 16) formulation is an exact algorithm to find the optimal solution at the cost of computational time and resources. The IP formulation is given by minimising $\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$ with the additional constraint of no sub-tours: $\sum_{i=1}^{n} x_{ij} = 1,\ j = 1, \ldots, n$ and $\sum_{j=1}^{n} x_{ij} = 1,\ i = 1, \ldots, n$ with $x_{ij} = 0$ or 1 if $x_{ij} \in X$. With D_{ij}, the distance between vertices i and j and x_{ij}, the decision matrix [5; 19]. X is the solution matrix of a collection of sub-tours breaking constraints, the maximum number of constrains can reach (2^{n-1}) with n cities. The values for the variables can be further constrained to be only integers, which is defined as the a more difficult integer linear programming (ILP) problem (problem 7) [12].

2.3.3 FUZZY LOGIC AND EXPERT SYSTEMS

Fuzzy logic and expert systems are linguistic approaches to control strategies and are based on human knowledge and logic decision making processes. A fuzzy system is a rule-based system and is often required where the system's mathematical model is not available or difficult to obtain.

Fuzzy logic can also be used for system classification as in the case for elevator traffic analysis. Fuzzy logic can be used to divide the passenger flow between trivial patterns identified by elevator experts, previous set of predefined patterns or passenger count samples collected over time. For example, passenger traffic distributions can be divided between up-peak, down-peak and interfloor passenger movements and then further divided between lunch hour times or any other event that can have an effect on the traffic patterns [20]. Difficulties in this approach are the inconsistencies in traffic flows throughout the building and the need to maintain and update the fuzzy rules continuously. Effective rules can be written, for example, when the elevator car has reached its passenger capacity at a certain floor all the elevator cars in that elevator group are send to that floor as soon they are available or further described by Koehler and Ottiger [20]: "If intensity is heavy, incoming traffic is high, outgoing traffic is low, and interfloor traffic is low, then traffic type is heavy up-peak". Only then will a specific control algorithm be triggered to send any idle elevators down to that landing floor.

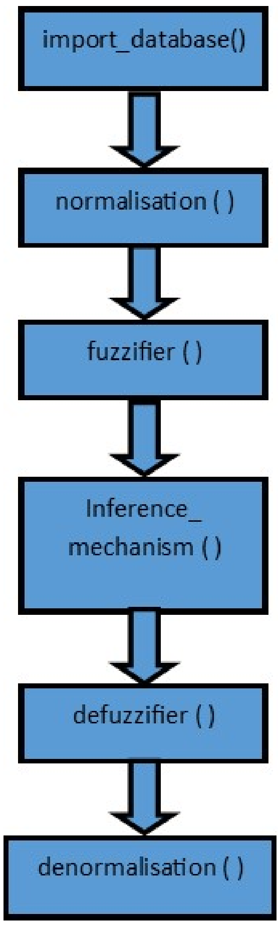


Figure 2.3: Fuzzification Flow Diagram.

Fuzzy logic is implemented by creating the membership functions by fuzzification and the member functions are then implemented by an inference mechanism to obtain the output by defuzzification, see Figure 2.3 [21].

2.3.4 GENETIC ALGORITHMS

Genetic algorithms are an evolutionary process that is derived from Darwin's theory of evolution in the field of biology. Evolution can be described as the change in the inherited characteristics of biological populations over successive generations, which creates diversity and genetic variation as a result of natural selection [22]. Darwin's research on evolution is based on three principles, namely: (1) Different individuals in a population have different morphologies, physiologies and behaviours, (2) differences give rise to different rates of survival and re-production in different environments and (3) there is a distinct correlation between the parents and their offspring in terms of their compositions and traits [23]. The different rates of adaptability to survive are often referred to as the differential fitness levels of the organism or biological structure. The better the fitness level, the better the chance of adaptability, reproduction and ultimately, survivability. Natural selection enforces the principle that if any entity have variation, reproduction and heritability, it will evolve [23].

From this area of research it brings forth genetic algorithms to implement this evolutionary process to real world applications, which are not necessarily biological. There is great need for certain processes to adapt and learn from its present state and adjust to be more efficient and coherent. But the changes are still micro-evolutionary, meaning the changes will not result into a new process or new species [3]. General uses for genetic algorithms are complex scheduling, routing, machine learning, searching, allocation and detection. Applications normally include optimisation problems that cannot be solved with traditional methods or existing computational powers. To refer back to evolution, genetic algorithms illustrates parts of a process as chromosomes and its genes as individual components to the program. Each chromosome must however represent one individual solution to the problem [3]. Genes are the parameters which can be optimised and changed in order for the single chromosome to give a better solution than before. The optimal solution is created by enforcing three operations, namely: (1) Selection, (2) crossover or mating and (3) mutation. Selection is based on the fitness levels of each chromosome and the privilege to mate. The mating is then accomplished by taking certain sequences of genes from each parent chromosome and divide them between offspring chromosomes. Mutation refers to the introduction of variation by either alteration, modification, transformation or metamorphosis of a chromosome by adding random sequences of genes. However, the higher the mutation rate the lower is the adaption rate for the offspring, which will negate the effect of optimisation and survivability.

Lotz [24] makes referral to three existing genetic programming software applications that can be used to model process systems. The main purpose of these applications is to solve complex regression problems not likely to be solved by hand. Genetic algorithm's main purpose is to determine the relationship between the independent and dependent variables and to search for the best combinations to ultimately reduce them. GA's can also be used to transform the independent variables in the most appropriate optimisation models, namely the Genetic Algorithm for Regressors' Selection (GARS) and the Genetic Algorithm for Regressors' Selection and Transformation (GARST) algorithm [25]. Lotz [24] first application makes reference to the GPLAB toolbox [26] that generates models by using mathematical operators, Boolean functions and operators. The GPLAB toolbox can compare parameters to other parameters, functions of a parameter or other numeric values. The second application that is made reference to, is the GP250 developed by Swart and Aldrich. This application generates models by only using mathematical operators and is quite resource intensive, being able to compile between 100 and 400 individuals per generation and can run between 100 and 400 generations which comes down to roughly 10 000 to 160 000 individuals [24]. Discipulus^{TM} is the third application Lotz [24] makes reference to, which makes use of binary code. A lot more individuals could be compiled by this applications with the same resources than the previous two GA applications, roughly in the range of 25 million to 26 million individuals [24]. These applications illustrate the potential of using genetic algorithms in real world applications and the pure computational power of these implementations.

The adaptability and search optimisation of genetic algorithms can be illustrated in the attempt to solve the well-known Traveling Salesman Problem and the Vehicle Routing Problem. As suggested by Jana Koehler and Daniel Ottiger [20], genetic algorithms can also produce better elevator dispatching solutions with a stochastic search approach rather than those generated by a predefined set of rules as in expert systems or with fuzzy logic dispatching methods. The full potential of this type of dispatching algorithm is still unclear and the right combination between mutation, mating and selection methods present a challenge in this domain [20]. However, it is proven that with GA the number of steps are considerably reduced and can be executed much faster by limiting the number of generations. The general flow of the GA is summarised by Figure 2.4, as described by Potvin and Thangiah [18].

2.3.5 NEURAL NETWORKS

Similar to Genetic algorithms we will look to nature to find more optimal methods for computer control applications to perform certain tasks and to combine these methods with its superior computational abilities. The biological neural network of the human brain is simulated and represented by the

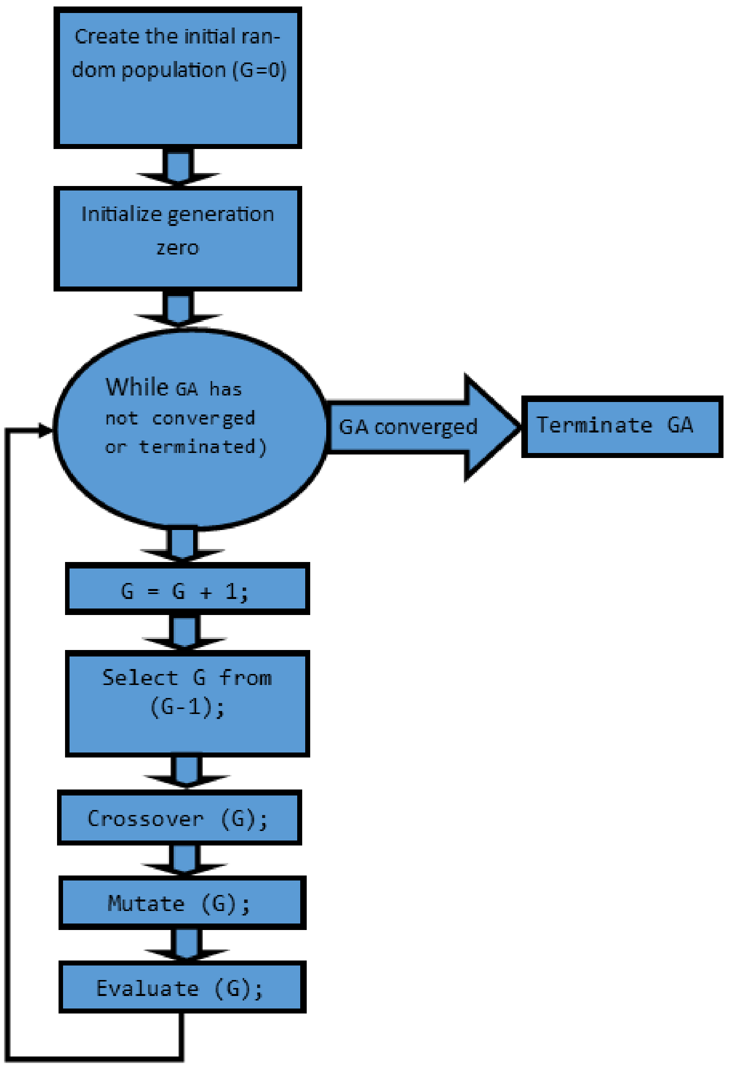

Create the initial random population (G=0)
Initialize generation zero
While GA has not converged or terminated)
GA converged
Terminate GA
G = G + 1;
Select G from (G-1);
Crossover (G);
Mutate (G);
Evaluate (G);

Figure 2.4: Genetic algorithms (GA) General Flow Diagram.

artificial neural network (ANN). The human brain is represented at the basic level, where multiple biological neurons are connected together and signals are received via its dendrites (received from the end nodes of another neuron), which in turn activates or fires the respective specific neurons. When a neuron is activated it will transmit the signal to other neurons via its axon [3]. A synapse is the space between the end nodes of the one neuron and the dendrites of the next neuron. The axon and the dendrites represent the "input" from a software perspective and the "output" is represented by the synapses of the neuron, see Figure 2.5. The neural activation process is in fact only a zero or a one, which means that a combination of neurons can be used to represent any basic level programming or function.

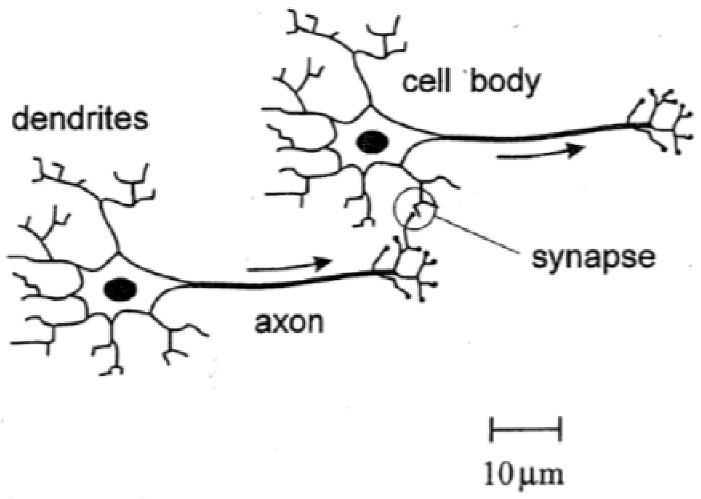


Figure 2.5: Schematic Illustration of Two Biological Neurons [2].

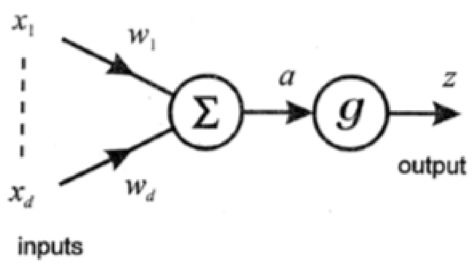


Figure 2.6: McCulloch and Pitts Model of a Single Neuron [2].

The origins of the use of neural networks date back the 1940's with research from McCulloch and Pitts [2] who modelled "a single neuron that forms a weighted sum of inputs $x_1, \ldots, x_d$ given by $a = \sum_i w_i + w_0$ and then transformed this sum using a non-linear activation function $g()$ to give a final output $z = g()$", see Figure 2.6. Two common attributes associated with every neuron are the threshold and the weights between them. "An incoming signal will be amplified, or de-amplified, by the weight as it crosses the incoming synapse and if the weighted input exceeds the threshold, then the neuron will fire" as stated by Heaton [3].

In the late 1950's Rosenblatt adopted McCulloch's model into the perceptron and developed the perceptron learning algorithm [2]. Research contributions continued slowly until the 1980's where the physicist Hopfield developed his own learning algorithm based on error backpropagation. The Hopfield network is defined as a single layer, where every neuron is connected to each other as illustrated in Figure 2.8. The network is classified to be auto-associative, which means it returns the same pattern it recognises [3]. In a Hopfield network, the neurons do not have connections to itself, thus a four neuron network has only 12 connections and also does not contain a threshold value as in the case with some other neural networks. Each connection and its associated weights can be presented by a 2 dimensional weight matrix [3]. The weight matrix in Table 2.4 will recall the patterns 0101 and 1010. When such patterns are presented it summates all the weights that have a 1 in the input pattern. For an input of 1010 we will have an output vector with the following values: $N1 = (-1) + (-1) = -2$; $N2 = 0 + 1 = 1$; $N3 = (-1) + (-1) = -2$; $N4 = 1 + 0 = 1$. With a threshold or activation function the neuron will fire if the output is above a certain value. Hopfield classified the activation function to be any value above zero, thus only $N2$ and $N4$ will fire in this case, which resulted in the input pattern as expected.

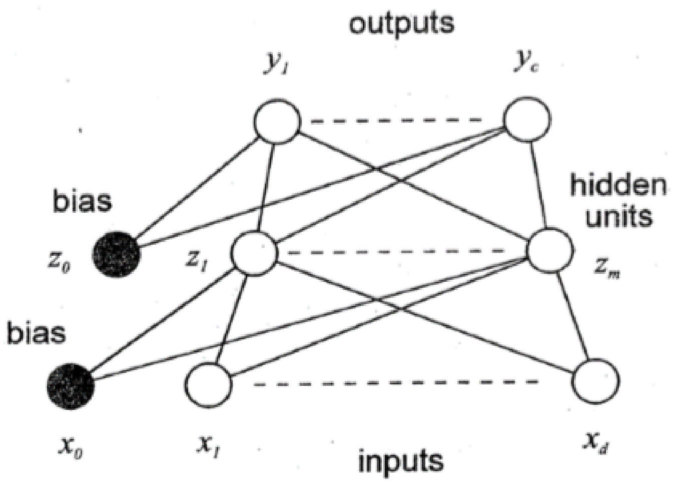


Figure 2.7: A Multilayer Perceptron Neural Network Having Two Layers of Weights [2].

Table 2.4: 2 Dimensional Weight Matrix to Represent Each Connection and its Associated Weights [3].

	Neuron 1 (N1)	Neuron 2 (N2)	Neuron 3 (N3)	Neuron 4 (N4)
Neuron 1 (N1)	0	-1	1	-1
Neuron 2 (N2)	-1	0	-1	1
Neuron 3 (N3)	1	-1	0	-1
Neuron 4 (N4)	-1	1	-1	0

The feedforward neural network is where its neurons are only connected to the next neuron layers and do not make any contact with the neurons behind its current position, as illustrated in Figure 2.9. Hidden layers are often

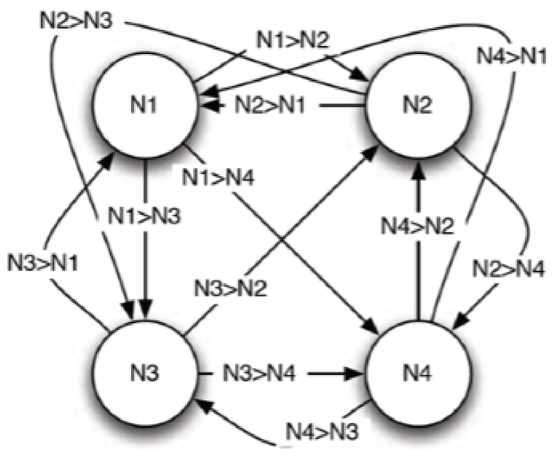


Figure 2.8: A Hopfield Neural Network with 12 Connections [3].

utilised to improve the outer results but at the cost of making the network more complex. The feedforward neural network is often implemented together with the Backpropagation training method to compare the anticipated output to the input with an error calculation function. The result obtained from the error calculation is then used to adjust the weights of the output layer back to the input layer. Backpropagation is classified as supervised learning, because the anticipated output values are required for the error calculation to be conducted. Activation functions are used to scale the neural network output layer into expected ranges or groups. Various activation functions can be used or created, but most commonly used today are the sigmoid function, hyperbolic tangent activation function and the linear function [3].

Problems that are well suited to the adaptive abilities of neural networks are usually when there is a pattern recognition or a classification requirement by the system. Neural networks are trained to recognise certain patterns or classifications by providing it with training samples and when the same or similar samples are provided, it will attempt to provide the expected output related to that specific pattern or group. Optimisation and prediction problems are also effectively solved by neural networks like optimising the traveling salesman problem and predicting the financial stock markets [3]. Problems that are not well suited to neural networks are usually stepwise systems which is compiled with logic statements and do not change over time [3]. Systems that are not well suited are where you are particularly interested in how the output was generated and the specific flow of the process, because the configuration and steps of neural networks are often hidden to the user.

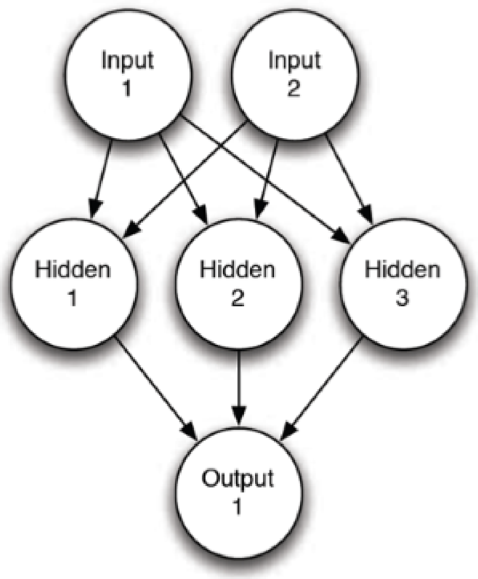


Figure 2.9: A Typical Feedforward Neural Network [3].

2.3.6 SELF-ORGANISING MAP

The self-organising map also form part of the neural network architecture and was developed by Tuevo Kohonen. The self-organising map does not return a pattern like the feedforward backpropagation neural network, but rather produce a single binary value as output from the winning neuron. By implementing this network there is only one output neuron that fires, thus the need to scale the output neural layer by means of an activation function becomes unnecessary and is not required anymore. The major benefit to the self-organising map is that it is an unsupervised training mechanism, where the output data is not required to train an effective network and thus hidden neuron layers are also not utilised. The input pattern, however, first needs to be in a binary form, that is between -1 and 1, thus it first progresses through a normalisation step before the input neuron layer receives the normalised values.

Problems that are well suited to the self-organising map includes systems that require data classification or cluster identification. The network train by adapting to the input pattern by gradually adjusting the respective connection weights [5]. When the traveling salesman problem and the vehicle routing problem are solved by this network, a ring is initially defined and gradually modified until it gets close to a city to create a return path or a tour [5]. In this way certain input patterns (city order) are grouped or clustered together, where the neighbouring cities are also more inclined to form part of the winning ring to a reduced extend [5].

2.3.7 SIMULATED ANNEALING

The term annealing refers to the metallurgical process of heating solids up to a point or colour and then slowly cooling it down until it crystallises [3]. When integrated chips (IC's) are manufactured the infusion doping of ions to the surface causes dissociation of the molecules into its atoms as result of the rapid heating [4]. To avoid this occurrence different annealing techniques are introduced, to increase the anneal temperatures without surface dissociation. But we also want to reduce the anneal time to avoid significant dopant diffusion, resulting in deeper junctions [4]. The different techniques include furnace and spike annealing to reach the 45 nm CMOS technology requirements and depth, but for ultra-shallow 2 nm junctions, flash and laser annealing can be utilised. Flash annealing uses high voltage pulses to achieve the high temperatures in the millisecond time range and laser annealing manage to reach these high temperatures in the micro to nano-second time range. With higher temperatures the atoms have more energy to settle down and with a higher level of freedom, see Figure 2.10.

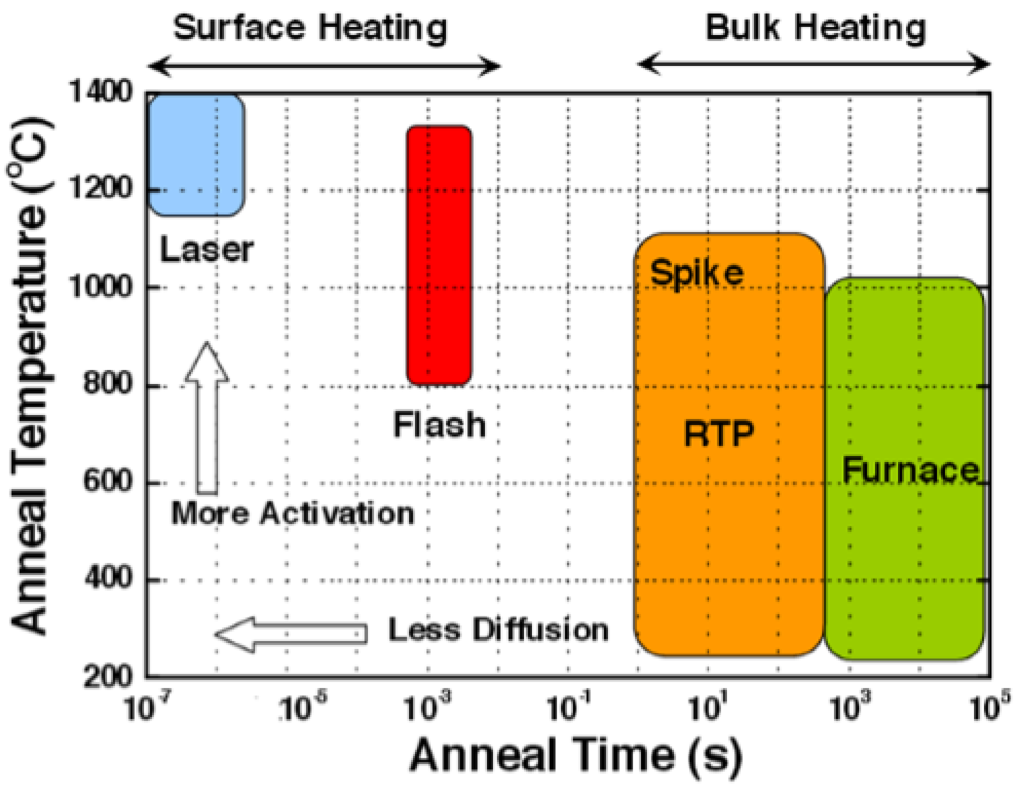


Figure 2.10: Temperature-time Ranges of Various Conventional and Advanced Annealing Techniques [4].

Simulated annealing in an algorithmic implementation attempts to emulate the metallurgical process, described in the previous paragraph, where you begin with very high temperatures and a wide range input randomisation (increased freedom). For every pre-defined number of cycles the temperature is reduced and the range of input randomness is decreased. For every cycle the set of inputs that produce the best results, up to this point, is retained. The process continues until the lower temperature boundary is reached or when the best result has not improved for a number of continuous cycles.

Problems that often benefit from simulated annealing are where you have a specified number of inputs or variables that you want to solve for an arbitrary equation or when you want to simplify its algebraic equations or system configuration. Simulated annealing can be used to randomise the weights of a neural network within the range the weights are limited to [3]. The Traveling Salesman Problem can also be solved by randomising the order of the cities by the simulated annealing process as required until the best route is found.

2.3.8 ELASTIC NETS

The elastic net is modelled in much the same way as the self-organising map in terms of the initial ring that is created for city clustering, however the way they update the coordinates of the points on the ring differs to each other [5]. An elastic net algorithm does not form part of the neural network architecture, but is rather an iterative procedure. The purpose of the iterative procedure is to minimise the length of the ring and the points on the ring. Initially there are more points on the ring than the number of classified cities until each point converge to a singular city. Let X_i be the position of the i^{th} city and Y_j be the j^{th} point on the ring. The ring points are updated with the following equations, where β and α are constant parameters and K is the scale parameter [5]:

$$\Delta Y_j = \pm \sum_i w_{ij}(X_i - Y_j) + \beta K(Y_{j+1} + Y_{j-1} - 2Y_j), \ j = 1, \dots, M \tag{2.3.1}$$

$$w_{ij} = \frac{\phi(d_{X_i Y_j}, K)}{\sum_k \phi(d_{X_i Y_k}, K)} \tag{2.3.2}$$

$$\phi(d, K) = e^{-\frac{2d^2}{2K^2}} \tag{2.3.3}$$

$d_{X_i Y_j}$ is the Euclidean distance between the city i and the point j on the ring and w_{ij} is weight of city i to the point j. The first term in Equation 2.3.1 drives the point towards the city and the second term tries to keep the neighbouring points together. In order to solve the TSP the global minimum of the derivative of the above equation needs to be calculated $K \to 0$ and $M/N \to \infty$, where the energy function is defined as:

$$\Delta Y_j = -K \frac{dE}{dY_j} \tag{2.3.4}$$

$$E = \alpha K \sum_i ln \sum_j \phi(X_i Y_j, K) + \frac{\beta}{2} \sum_j (d_{Y_j Y_{j+1}})^2 \tag{2.3.5}$$

It is stated that from an elastic net algorithm the average number of iterations to converge in comparison to simulating annealing is about the same [5]. But when you allow each point on a ring to match more than one city, when the cities are sufficiently close, the elastic net can be superior to the simulating annealing algorithm [5].

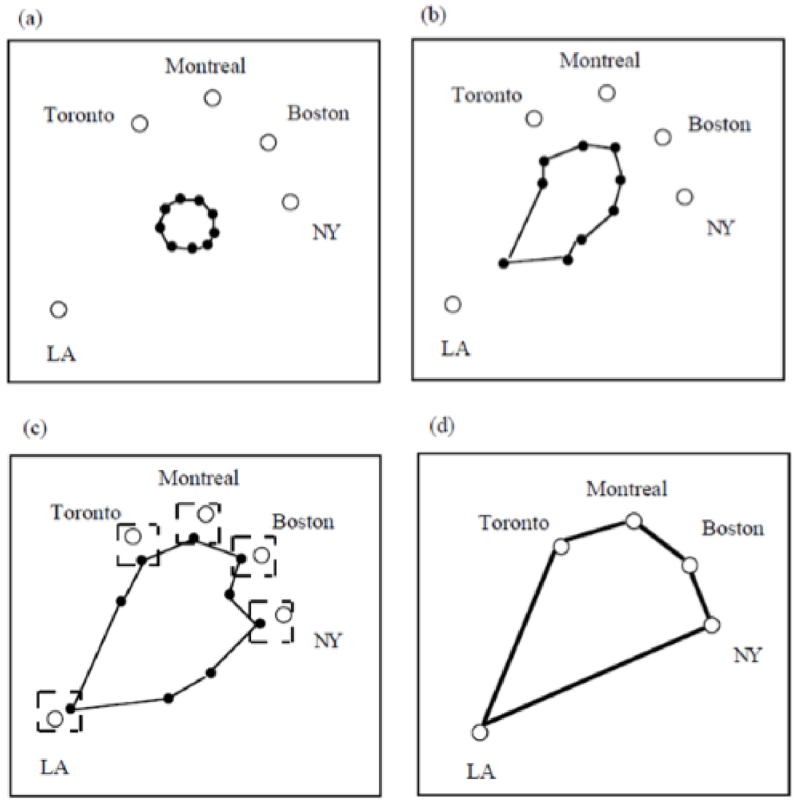


Figure 2.11: Evolution of The Elastic Net Over Time [5].

2.4 SUMMARY AND CONCLUSIONS

This chapter presents a brief introduction to elevator control possibilities and provides some background information on the history of elevator technologies. Mention is made to online vs. offline strategies and the implementation thereof. Various machine learning concepts and artificial intelligence philosophies are introduced throughout this chapter with the intent of implementing some of them in an elevator application. Specific reference is made to Neural Networks, Genetic Algorithms and Simulating Annealing.

Chapter 3

ELEVATOR CONTROL: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

3.1 INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is the discipline of machine intelligence and can be divided into symbolic and computational intelligence. Symbolic intelligence can be seen as the origin of machine logic that was created by man, in order to establish a data representation and reasoning process for developing different AI applications [27]. The first instances of AI were to process information in a vast and effective manner, but by imposing more complex problems for AI machines to solve, the need for more advanced computational abilities were required. We require machines to be able to learn, anticipate and adapt to the immediate environment and not just to follow symbolic logic as previously thought.

Intelligence has a strong link to the term cognition, which is defined as the process whereby sensory input or information are processed, knowledge is applied and preferences are altered. In human psychology cognition refers to our understanding and processing of information relative to affection, motivation or preference [27]. Thoughts and awareness also have close relations to cognition and intelligence. A human being has different thought processes which he/she uses to reflect to the immediate and perceived reality, for example perceptual, imagery, inspirational and abstract though processes. The easiest way to replicate any of these processes would be with attempting to create abstract thoughts in a machine. Abstract thoughts in humans are possible where there is an interaction with an abstract concept and some sort of a language or symbols to produce abstract thinking. Therefore machine thinking may be possible with a fully established symbolic logic structure and an abstract concept definition. In terms of awareness or attention in the cognition process, it plays an important role to ensure that the correct action or changes are made when it is required. As with humans, machines operate mostly serially. Thus it is important to execute each required function at the exact time it is needed and to exit the function as soon as possible. The ability to loop and force the machine to execute its functions continuously must be reduced, if we are ever to create some kind of consciousness.

3.2 INTELLIGENT SINGLE ELEVATOR CONTROL: GENERAL PHILOSOPHY

The intelligent single elevator control philosophy is based on the adoption of the widely known Traveling Salesman Problem (TSP). In this case the traveling salesman will be represented by the elevator car and the cities will be the different floors. The main variations are for each floor to be visited only once in every floor order instance and the starting floor can be adjusted. The position coordinates on the x- and y-axis which normally dictates the distance calculations between the cities will be variables in this case and used in much the same way. The variables will be used to get the best order of floors with the shortest "distance" or cost calculation from one floor to another. Typical types of distance calculations are given by [27]:

The Manhatten distance:

$$d_{ij} = \sum_{k=1}^{N} \mid V_{ik} - V_{jk} \mid \tag{3.2.1}$$

The Euclidean distance:

$$d_{ij} = \sqrt{\sum_{k=1}^{N} (V_{ik} - V_{jk})^2} \tag{3.2.2}$$

The Minkowski distance:

$$d_{ij} = \sqrt[q]{\sum_{k=1}^{N} \mid V_{ik} - V_{jk} \mid^q} \ , \ q > 0 \tag{3.2.3}$$

Where V_{ik} and V_{jk} are the respective values of the k^{th} feature of case i and j, also q is used as a variable generaliser instead of 1 or 2 as in Eq 3.2.1 and 3.2.2.

The three main objectives for a vertical transportation system can be categorised into the following sections:

- Travel cost.
- Service level.
- Throughput maximisation.

These concepts will be thoroughly investigated throughout this thesis and implemented with various variations throughout this elevator simulated model. The concepts are all dynamic variables and can be set according to any predefined criteria and are predominately linked to information that are received from an elevator input system or directly from an updated database.

Travel costs are mainly categorised as power consumption and waiting times. The power consumption of any electrical system has become very important in present times, due to increasing electricity prices and limited energy resources. The main objective in developing a more energy efficient control system is to reduce overall energy consumptions, but not to reduce the functionality and effectiveness of the application. We will be developing an Intelligent Elevator Controller (IEC) that can make more intelligent decisions that take power consumptions into consideration. The following is a summary list of possible Influences and/or dependencies in relation to power consumption:

- Elevator motor and the controller's technology and efficiency ratings.
- Load (amount of weight traveling per instance).
- Number of stops.
- Direction.
- Counterweight settings.
- Standby and running times.

Further by measuring various waiting time parameters we will be able to compare and optimise elevator functionality and effectiveness of the controller. The IEC are to take waiting times into consideration when the optimal floor route is calculated, but also to be able to factor in probability and power consumption in every decision. The following is a summary list of possible influences and/or dependencies in relation to waiting times:

- Elevator configuration and technology including motor rated speed.
- Equipment and passenger delays.

- Traffic distribution at landing floors and inside the elevator car.
- Number of stops.
- Control philosophy.
- Maintenance.

An elevator application's main purpose is to transport people, which introduces an important category, namely the **service level** of a system. Service levels are directly related to the passenger's satisfaction and perception of an efficient elevator system. It is a subjective term and can vary from person to person and is related to waiting time parameters, comfortability, safety and user-friendliness.

Throughput maximisation can be generalised as the maximum amount of passengers or goods that can be serviced in a time period. Usually throughput maximisation is limited to various compromises as a result of the degree of influences from other system objectives. Also the system is often controlled by physical and safety (legislation) constraints as well.

In order to develop an intelligent elevator controller to satisfy the above objectives we need the system to behave less dynamically. The actual traffic flow will always make the system behave in ununiformed patterns, but the system can be more prepared to deal with the randomness to a certain degree. Thus **probability** plays an important role in any control system that deals with input information, which change continuously and where there is limited time to respond to these input changes. We require that the IEC to have a few prediction functions, in order to reduce ambiguity and minimise input uncertainties. The following is a summary list of possible influences and/or dependencies in relation to probability:

- Past traffic distribution density and analyses.
- Floor importance (offices, manager's floor, main landing floor).
- Passenger behavioural information.
- Working environment dynamics.
- Building configuration.

3.3 ELEVATOR MODE PERSONALISATION

Normally a building owner has little influence on how each elevator is operating throughout her/her premises and has a limited perspective regarding any user control functions. In the past user requirements were compiled and made part of the control design requirements before the commissioning stage of any new elevators. However any variations to the initial specifications are not easily incorporated and can be very costly, but priorities and set expectations do change and need to be catered for. With a more developed Intelligent Elevator Controller; the different operating modes can be influenced by the user with adjustable main and secondary objectives. The IEC should be able to process the user input requirements together with the criteria weights to be able to adjust the control parameters, with the aid of a graphical user interface (GUI). Alternatively to create a more user friendly operating system, the system constraints and operations shall also be able to adapt and adjusted heuristically throughout the operating cycle.

General elevator adjustable user modes can be defined by Table 3.1. The suggested modes shall also be used as a feedback mechanism to provide the user the required information about the current state of each elevator. For example mode 0 (out of service) can either be implemented by the user to take the elevator out of service if certain conditions are met; like car is empty and stationary or the elevator mode can provide feedback that it is not in service at the moment. Mode 1 allows the controller to prioritise power consumption reduction when the cost function is implemented to generate the optimal elevator route. This mode can also be defined as the secondary objective when high passenger demand is experienced and throughput maximisation takes priority.

Waiting times in terms of personnel being transported or goods being delivered should be defined differently. Thus modes 2 and 3 differs in terms of the application requirements; if goods are predominantly being transported by certain elevators then some settings can be adjusted accordingly. For instance the doors can be kept open for longer periods; to allow for longer goods transfers in and out of the elevator car and fewer stops between origin and destination floors are allowed; to reduce delivering times. It may also be feasible to reduce the speed of the elevator car to a minimum if critical loads are to be transported or to allow specific cars to be accessible only trough valid authorisation; to avoid misuse of the goods elevators by passengers.

Some elevators can be designated to be used in emergency situations and is set to the emergency mode, either primary or as a secondary function. These elevators can be fully equipped with extra medical equipment or firefighting apparatus and can be designated to specific floors like the intensive care unit (ICU) level at a hospital. The balanced mode is generally the default setting that will optimise waiting time, power consumption and throughput maximisation by an efficient dispatching control system. Some elevators can be put on standby mode, which means that the elevator car is still utilised in high traffic demand states, but otherwise not in use if traffic is low. The elevator can be put on override mode if the building has an automated control system, which will deactivate any external user control interfaces. The elevator can then be put on a pre-programmed route or as required by an external input system as in the case with a process manufacturing line. Lastly the maintenance mode can be utilised in order to minimise elevator usage on a particular elevator until maintenance is concluded. The maintenance mode can assist in prolonging in-service time until moderately damaged cables can be replaced or to avoid a problematic motor from overheating until its replaced.

Table 3.1: User Personalised Elevator Control Modes.

Mode	Representation and Priority Criteria
0	Out of service
1	Power consumption
2	Waiting time: personnel
3	Waiting time: goods
4	Emergency
5	Balanced
6	Standby
7	Override
8	Maintenance

The notion behind an intelligent and personalised elevator controller is to be able to choose between different modes as soon as the user requirements change. When a building has a number of elevators, the user should be able to define each elevator individually as required to suit the application and to optimise each elevator more effectively without costly controller reconfigurations.

3.4 INTELLIGENT ELEVATOR GROUP CONTROL

3.4.1 INTRODUCTION

The objective for an efficient controller is to control each elevator car in the group in such a way that will provide the best overall results in terms of cost minimisation as well as balancing power consumption and throughput maximisation. In terms of an artificial intelligent controller it is also important that there is a cognisance between the elevator cars and that each decision also benefits the rest of the car movements.

3.4.2 VRP: TSP EXTENSION

After we have developed an efficient TSP algorithm with a single elevator car configuration, we can easily utilise the same algorithm if more cars are added to the configuration. The optimal calculated route becomes the main route, which are then divided into smaller tours and allocated to each available elevator car to complete. Thus the Traveling Salesman Problem is extended into the so called Vehicle Rooting Problem (VRP), where each car is dispatched by the controller as required and receives a designated route to follow. Each car in the group executes its own distinct route from the main tour separately from the other cars. It can then be further extended to allow cars to extract smaller tours from other active cars which have not completed their allocated tour. Each tour have at least a start node, where passengers are picked up and an end-node, where passengers are delivered to their destinations. One of the constraints is to disallow an elevator car to extract end-nodes, but should only extract smaller tours where passengers are still to be collected (start-nodes).

The main route can be split up into smaller segments by a number of techniques for instance by clustering methods such as by the basic k-means, with reference to Section: 7.2.1. The floors in the pre-defined route can be grouped together based on physical proximities and the number of groups can be optimally calculated. The optimal number of clusters should be equal or less than the number of elevator cars in the group. Further it should then be proven that by dividing the main route into smaller tours it is more optimal than the initial calculations. Golden *et al.* [28] propose sequential search algorithm with gain criterion to proof such divisions. Sequential search algorithm is utilised to check overall gain if one node is deleted from another and iteratively repositioned to the existing nodes [28]. Sequential search relates directly to edge-exchange and node-exchange neighborhoods to decompose routes into smaller tours [28].

3.5 VARIATIONS AND CONSTRAINTS

Further the TSP and the VRP can be developed with more complex variants to the basic definitions. These variants can have 3 additional constraints for each elevator car and should be individually adjustable, namely load/capacity constraints, time window constraints and passenger type constraints.

Load constraints have to do with the rated weight/load and the maximum available floor area of each elevator car. As required by South Africa National Standard (SANS) 50081-1 (EN 1-1:1998) clause 8.2 Table 1.1 with the rated load per passenger estimated at 75kg. The elevator dispatching algorithm should also take the ratio between the amount of passengers at a landing floor over the available car capacity at a set time period into account. This ratio defines the maximum amount of trips required to service all the passengers at that floor, which in turn will influence the waiting time between consecutive trips and also transportation cost to that floor.

Time window constraints plays an important part in waiting time calculations. One of the primary objectives for the elevator controller is to minimise the overall waiting time, which can be achieved by neglecting individual waiting times by prioritising larger passenger groups. By means of a time window constraint per passenger, we will also prevent that passengers are overlooked by the dispatching algorithm. The ratio between the time between consecutive and occupied floor visits and the time window constraint need to be optimised against the overall minimum waiting time. This creates a complex relationship between the traffic flow rates and delivery route travelling times.

Passenger type constraints, also known as site-dependent VRP (SDVRP), are where there are compatibility relations between passengers and the elevator cars. For example passengers with special ability requirements can be routed to use specific elevators which cater for their needs specifically and also passengers with goods are recommended or obligated to only use certain classified elevators. The passenger type constraint can also be used to implement preference between passengers, for instance some elevators will can give preference to either permanent employees over visitors or vice versa, or prioritising VIP's over the normal work force, depending on the organisation's user requirements.

A further variation to the general TSP and VRP can be investigated when the floor order constraints are reduced by allowing the various algorithms to have the possibility of visiting the same floor more than once in the same projected floor order list. The length of the floor order list is now a pre-defined number of floors and not limited to the total amount of floors the elevator is able to visit in the building. With this variation the computational requirements becomes much more complex, depending on the length of the required floor order list.

3.6 INFORMATION RECONCILIATION

The IEC simulation requires continuous passenger information from all floor landings and elevator cars, to be able to provide an optimal floor order for elevator dispatching at the exact time it is required. The data required by the controller can both be obtained and processed periodically or whenever an external (or internal) event has triggered the required data to be collected. When the data is collected periodically, let's say every few seconds, the controller would refresh and reconsolidate all the available passenger information and elevator car positions. Then the data is fed to the respective algorithm classes or methods in order to get an optimal or preferred floor order list. When the controller is faced with an increase in computational requirements, the controller also has the possibility to make use of an action listener philosophy. This means that instead of executing the algorithms periodically we can wait until a trigger or an event has occurred, which will indicate a possible change in the previous collected data. When the action listener philosophy is implemented instead of the polling philosophy; the total computational time can be reduced considerably. Any changes to the collected data will require either a total re-calculation of the planned floor order or only an adjustment to the existing planned route. The action listener philosophy also provide the additional benefit of a real time simulation unit and will more likely be implemented in a real world elevator controller over the other philosophy. An event can be anything from a change in passenger information (service requests), elevator car locations, fault alarms, warning signals and safety device triggers.

3.7 SUMMARY AND CONCLUSIONS

We are investigating and employing different machine learning techniques throughout this chapter and the rest of the thesis, in order to improve the computational and symbolic intelligence of the elevator control system. Further we are attempting, throughout the development process, to invoke an artificial consciousness or awareness in the elevator control system. The main focus is to obtain the best possible decisions with the available resources and limitations, but also to create an AI system that can provide results based on its own artificial intellect and not necessarily with pre-programmed responses.

Chapter 4

ELEVATOR POWER CONSUMPTION

4.1 POWER CONSUMPTION PHILOSOPHY

In order to improve the elevator controller's decision making process, we need to include power consumption data from the overall elevator configuration into the development of a more intelligent controller. To optimise power consumption, the focus is shifted from the overall passenger waiting time reduction to a reduction in transportation cost. There are a few philosophies or proposals, which can be implemented throughout the decision making process in an attempt to reduce overall power consumption. However the overall controller functionality and computational resources available should not be neglected or impacted negatively.

The general philosophy is to create a general energy model for our elevator configuration, which are based on accurate and relevant energy consumption estimations. The power consumption data can be theoretically calculated or obtained through inductive reasoning with a well-established database, or by actual energy measurements. This energy model is then used to estimate the energy consumption for a chosen floor route and compare the estimation against other routes. This philosophy can be incorporated in the TSP as an additional variable, where the shortest route is calculated between the set variables, which are power consumption and waiting time in this case.

Other philosophies can include small alterations to the actual elevator car dispatching algorithm and decision making logic. For instance; if the floors which are located in close proximity of each other are serviced together, then the traveling cost can be reduced. Especially in the case of group elevators, where each elevator can be allocated to a designated portion of the floors to service. However deciding on which floors to include on a route is a complex and non-trivial problem. Also it may be worth looking into if the controller attempts to stretch the limitations set out be the waiting time philosophies. The result can be positive in terms that we are keeping to our waiting time limits and saving power in the same time. The opportunity arises where the elevator controller can dispatch a car to a floor at just the point where maximum passengers can be collected. Another scenario worth investigating is for the controller to maximise the ratio between available space in the elevator car and the amount of people waiting at the floor. This can possibly avoid having an elevator car traveling with almost no load and wasting power for single passengers, but still taking waiting limits into account. The controller can also be further limited in a more generalised way, for instance by reducing the amount of allowed visits to all floors throughout a time period, but still keeping to the maximum allowed waiting time per passenger. The main objective for these small changes is to minimise the amount of trips to the required floors, but still be able to transport all waiting passengers to their respective destinations in acceptable times.

4.2 ELEVATOR MODEL

The emphasis on total energy consumption shall now be shifted to smaller facets, where a more focused approach is required. An energy model must be established that indicate the influences various characteristics have on the total energy consumption and then this data can be combined with an established traffic pattern and actual loads to simulate an accurate energy model.

4.2.1 ELEVATOR CONFIGURATION

There are numerous elevator configurations that can be found in the industry at present, but they are mainly classified according to their main purpose, namely passenger, freights and dumbwaiter elevators. Elevators can also be classified as passenger-only elevators, because of inadequate lifting capacity or for traffic optimisation purposes as well as goods-only elevators, because of possible inadequate ventilation in the elevator car. Dumbwaiter elevators are small transportation boxes designed to carry lightweight freight, typically being used in hospitals and hotels. An elevator configuration is then further classified by its type of technology to transport the elevator car, namely traction, hydraulic or rack and pinion lifts.

In this thesis, we will however only be focusing on traction elevators, where traction elevators refers to the contact between a suspension system and a sheave. The suspension system is either ropes or cables with the number of strands configurations for variable strength requirements. The sheave configuration can be further divided between a system that uses a reduction gearbox

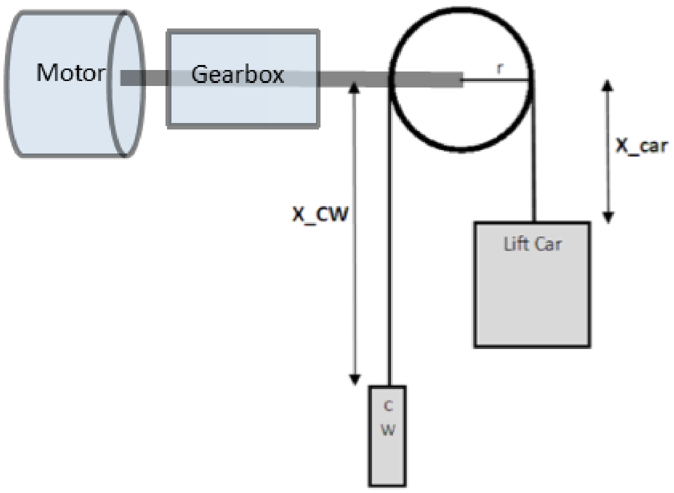


Figure 4.1: Theoretical Geared Elevator Model

to connect the motor with the suspension system or a gearless machine where the suspension system is directly attached to the rotor of the motor or via a ratio speed reduction rope pulley system.

4.2.2 THEORETICAL GEARED ELEVATOR MODEL

Elevators are optimally used in terms of power consumption when there is a perfect weight balance between the elevator car, counterweight and suspension system. In this occurrence the driving machine is only required to overcome the starting torque and to accelerate or decelerate the elevator car to its rated speed or to bring the elevator car to a complete stop. The mechanical and electrical losses included in the elevator configuration are mainly the motor copper and iron losses as well as friction and heat losses. The driving machine must continuously overcome these losses as well as the ever present gravitational forces in order to have a stable and responsive elevator control system.

Firstly we will be looking at the tension in the ropes created by the elevator configuration and ultimately the moment of inertia created by the load imbalances. If we neglect friction, compensation and spring abilities of the ropes the following equations are introduced by Newton's 2^{nd} Law:

$$F_{\text{car}} = m_{\text{car}}g - T_{\text{RopeTension}} \tag{4.2.1}$$

$$F_{cw} = T_{\text{RopeTension}} - m_{cw}g \tag{4.2.2}$$

There are 2 general forces applied to the sheave; one is due to the rope configuration and weight imbalances and the other is due to the applied or induced motor torque. With the distance ds being diameter of the sheave. Let's say $m_{\text{car}} < m_{\text{cw}}$ and the elevator car is required to move upwards, which means we require a nett torque in the anti-clockwise direction to overcome the load imbalances on the sheave and as a result accelerates the load in the required direction:

$$T_{\text{nett}} = T_m - (m_{\text{cw}} - m_{\text{car}})g\frac{d_s}{2} \tag{4.2.3}$$

The torque, which is an applied force to the elevator sheave, causes the sheave to turn in the direction of the nett applied force. The greater the torque on the sheave, the more rapidly the angular velocity of the sheave changes. According to Chapman [29] "the torque on the sheave depends on the magnitude of the applied force and the distance between the axis of rotation and the line of action of the force". The weight of the overall elevator configuration is not of importance here, because we are only interested in the load imbalances that occur. The compensating chains have the same weight as that of the main ropes and the counter weight is normally specified to be the same weight as the empty elevator car plus 45 to 50% of the rated car capacity. Thus an empty elevator car and a full elevator car will have the most effect on the required torque. The nett torque at the motor side with the inclusion of the gearbox reduction ratio r_g and the gearbox overall efficiency percentage n_f are:

$$T_{\text{nett}} = T_m - (m_{\text{cw}} - m_{\text{car}})\frac{gd_s}{2n_f r_g} \tag{4.2.4}$$

With the reduction or step-down gearbox; the required torque from the motor is effectively reduced with the mentioned ratio to overcome the load imbalances and to accelerate the shaft up to the rated speed of the motor. The gearbox efficiency have a negative effect on the generated torque and power, with the forward efficiency n_f often different than the feedback or reverse efficiency n_r. The overall inertia at the sheave can be given as the sum of the load inertia and that of the sheave and the shaft,

$$I_{\text{tot}} = I_{\text{shaft}} + I_{\text{sheave}} + I_{\text{load}}, \tag{4.2.5}$$

with

$$I_{\text{load}} = \frac{d_s^2}{4}(m_{\text{car}} + m_{\text{cw}}). \tag{4.2.6}$$

The total inertia being reflected back through the high speed motor shaft is reduced by square of the gearbox reducer ratio, thus:

$$I_{\text{motor shaft}} = \frac{1}{r_g^2}(I_{\text{shaft}} + I_{\text{sheave}} + \frac{d_s^2}{4}(m_{\text{car}} + m_{\text{cw}})) + I_{\text{reducer}} \tag{4.2.7}$$

In order to optimise the mechanical system we require the motor and the load inertia's to be matched in order to have a stable system and to have efficient power transfer through the shaft. If the load inertia is large the required motor torque is increased in order to match the overall system inertia and for the acceleration of the motor itself. The weight of the overall elevator configuration, including the elevator car, counterweight, cables, compensating chain and the transported load will have an effect on the inertia of the system. Torque and the overall inertia are related by $T = \alpha I$ where α is the rotational acceleration, thus we can conclude with the following equation:

$$T_m - (m_{\text{cw}} - m_{\text{car}})\frac{gd_s}{2n_f r_g} = \frac{\alpha}{r_g^2}(I_{\text{shaft}} + I_{\text{sheave}} + \frac{d_s^2}{4}(m_{\text{car}} + m_{\text{cw}})) \tag{4.2.8}$$

With convention being: upwards and counter-clockwise being taken as positive values and vice versa. When the weight imbalances are in the same direction as the required motor torque direction, the motor is being accelerating at an increased rate and will reach its rated speed in a shorter time period. However additional braking torque will be required once the speed must be reduced again in order to bring the elevator car to a standstill, and much of the generated power is dissipated as heat through a breaking resister in the drive.

The mechanical power of the motor can be seen as the generated torque times the angular velocity, with the motor slip taken into account: $P_{\text{mech}} = T_m w_s(1 - s)$. Most of the mechanical power spent during acceleration periods, where the motor slip is low and the optimal motor efficiency point has not been reached yet. At this period the generated torque is at its highest. With the elevators application the motor is required to go through constant stoppages and with load imbalances that varies almost every trip. It is to be investigated that in order to be able to reduce the overall power consumption, we require more information regarding the impact the amount of trips have on the system power measurements as well as the load imbalances and the traveling times.

Calculating angular kinetic energy: With the conservation of energy principle, we can calculate the angular sheave inertia during the car acceleration period at the load side by:

$$mgh = \frac{mv^2}{2} + \frac{Iw^2}{2} \tag{4.2.9}$$

Where the potential energy changes of the different masses are equal to the kinetic energy of the moving load and the angular kinetic energy by the sheave. When the elevator car is traveling at constant speed, there is no kinetic energy changes, with the potential energy being drawn or absorbed by the system [30].

Calculating overall system efficiency: The overall system efficiency can be estimated with the following equations by White [31]:

$$N = \frac{FV_c}{75n} \tag{4.2.10}$$

$$n = n_1 \cdot n_2 \cdot n_3 \tag{4.2.11}$$

$$F = (m_{\text{car}} - m_{cw} + m_{\text{rated load}})\frac{1}{C_M} \tag{4.2.12}$$

Where C_M is the suspension ratio, V_c is the rated car speed, N, is the rated output power in HP, n_1,n_2,n_3, are friction pulley performance rate, friction pulley benches performance rate and the worm screw performance rate respectively. Where the empty car weight together with weights from the cables and compensating chain can be estimated with [31]:

$$m_{\text{car}} = 1000 + 50 \cdot \text{rated number of passengers} \tag{4.2.13}$$

Calculating gearbox reduction ratio: The gearbox reduction ratio can be calculated by taking the ratio between the linear car rated speed and the translated rotational motor speed [32] with,

$$\text{ratio} = \frac{V_c}{\frac{V_m}{60}2\pi} = 9.55\frac{V_c}{V_m}, \tag{4.2.14}$$

and the rotational motion through the gearbox is multiplied by r_g and divided by the sheave radius to get radians, thus we have:

$$r_g = \frac{d_s V_m}{19.098 V_c} \tag{4.2.15}$$

V_c is the elevator car speed with the suspension ratio taken into account, in $m.s^{-1}$ and V_m is the rated motor speed in rpm.

4.3 MEASUREMENT AND VERIFICATION

4.3.1 M&V INTRODUCTION

The term namely, Measurement and Verification (M&V), refers to a process which validates the energy savings achieved by a project or a modification. Through the M&V process the actual "before" and "after" recorded data can be analysed and compared to the theoretical and simulated results. Majuba Power Station, where the elevator measurements have been taken for this Thesis, is conducting a full elevator upgrade for all 25 traction elevators. The elevator upgrade project includes new variable speed drives and new gearless machines. The full upgrade will not be covered by the scope for this Thesis, however we still conduct the "before" state results and attempt to predict the future power consumption improvements. The M&V configuration and procedure for energy metering is illustrated by Figure 4.2, from Eskom Corporate Technical Audit Department [6].

The actual recorded data can then be processed into the following result categories:

- Total energy consumption over a predetermined period (1 year, 1 month, 1 week, 1 day, outage period, non-outage period, peak hours 17:00-19:00).
- Standby power in kW, standby hours per day, standby hours per year, average trip running duration, number of trips, etc.
- Total energy consumption over one elevator cycle with various loads (0%, 25%, 50%, 75%, and 100%).
- Traffic analysis over a predetermined period.

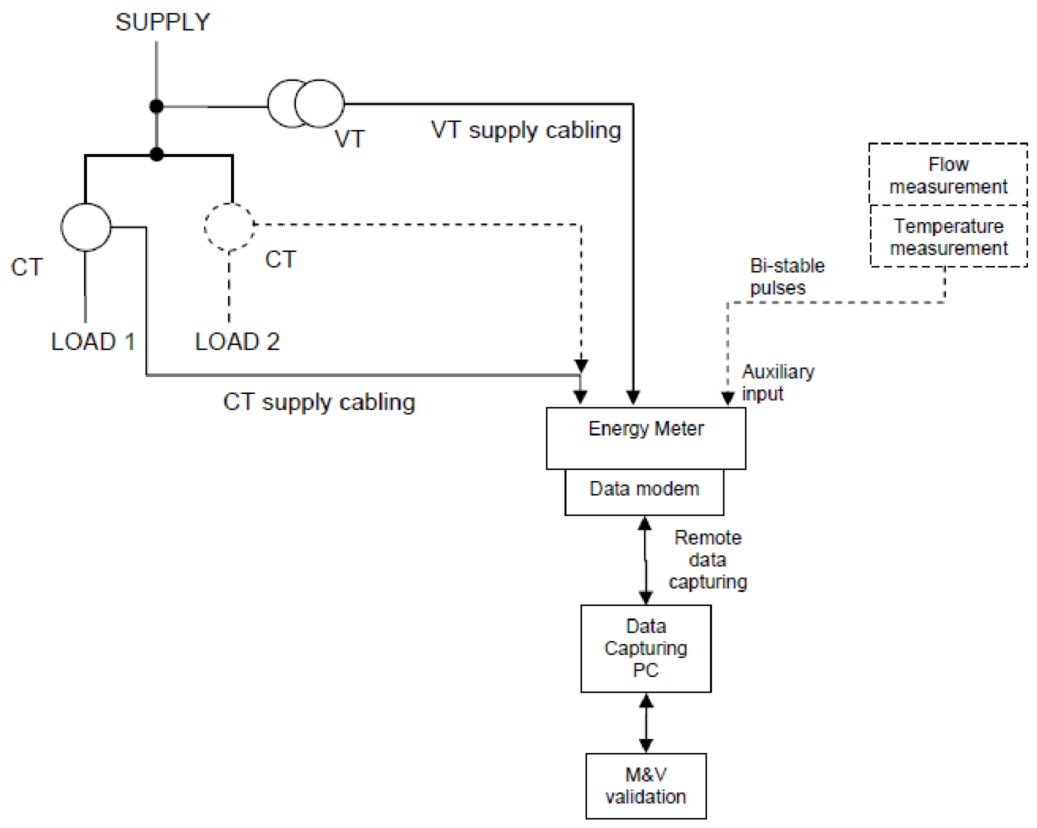


Figure 4.2: Measurement and Verification Energy Metering Process [6].

4.3.2 ELEVATOR CYCLE SAMPLING

Actual energy measurements can be done by various procedures, either by sophisticated energy measuring equipment with data recording abilities or by inexpensive ampere and voltage probe readings. Data sampling are carried out for the purpose of creating an accurate representation of the measured power signals. All the captured samples are stored into a database together with all respective timestamps and accompanied information, which will become useful later on. Required information is for example:

- Actual load, in terms of kg or number of people.
- Direction of travel.
- Actual elevator position during travel, including height above ground, starting floor, which floors were passed, destination floor.

The physical information in terms of the elevator car is captured manually for the purpose of this Thesis, alternatively accelerometers, mass meters, altitude meter and building management system feedback could have been utilised if available, to make the measurement process more dynamic and often more accurate.

All captured data is utilised for digital signal processing and data manipulation. The processed data is illustrated by means of graphical representation and can be further processed by various machine learning algorithms into meaningful information packages. The main purpose for collecting the various samples and processing it into useful information is to improve the decision making process of the elevator controller. The objective is to absorb all the obtained information (training data) to develop an approximate energy model and finally incorporate the model into the controller control algorithms and decision making processing unit. The energy model shall provide the required results for all expected elevator travels together with the experienced loads.

4.3.3 DATA SAMPLING PROCEDURES

The OMICRON CMC 256 Plus was part of the monitoring equipment used for data sampling. This equipment is normally implemented to test protection devices and for high precision calibration of measuring devices including energy meters, power quality and phasor measurements devices. For the purpose of this Thesis we used this equipment as a multi-functional multimeter and transient recorder. The OMICRON device generates digital output from the analog input signal samples, through its internal digital signal processing (DSP) capabilities. The OMICRON CMC 256 Plus also provides additional error correction algorithms and an accuracy oriented amplifier to improve test signals with low amplitudes [33]. Its accuracy is given with an error less than 0.06% of the amplitude readings and an input impedance of $600k\Omega$ and $5pF$. Sampling frequency can be set to 28.44kHz, 9.48kHz, and 3.19kHz. The software used in the application is the powerful Test Universe, version 3.0.

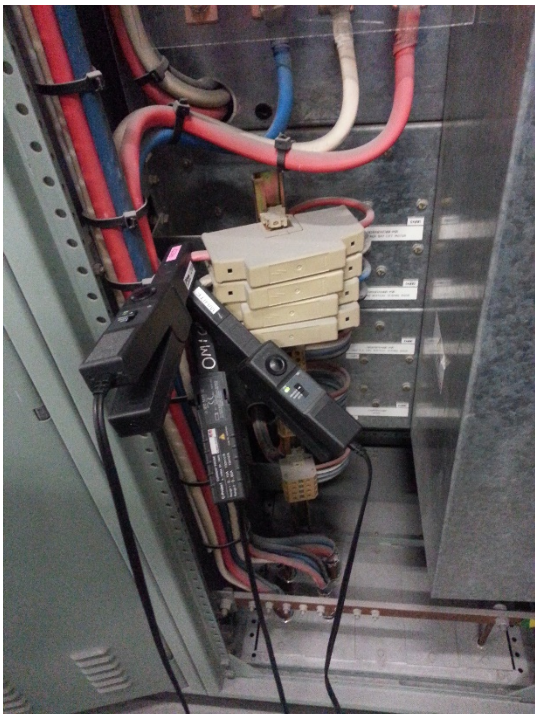


Figure 4.3: Data Sampling with the OMICRON CMC 256-Plus Using Analog Clamp-on's.

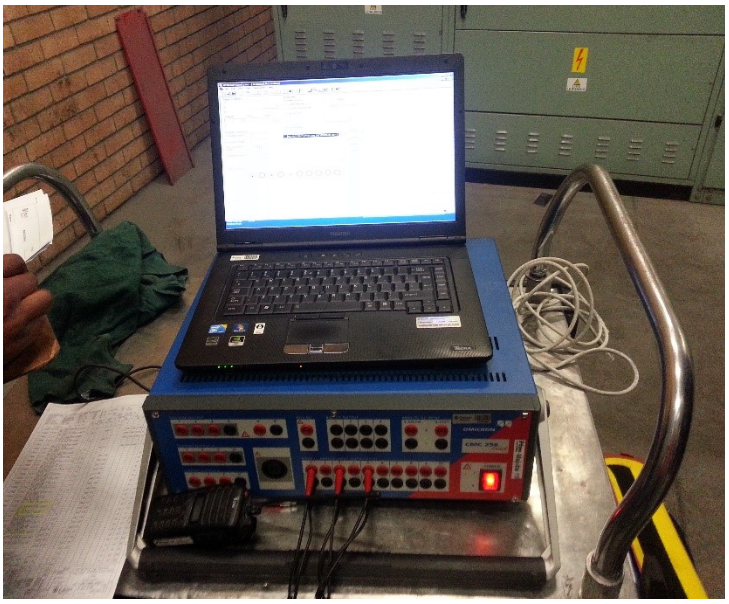


Figure 4.4: Data Sampling with the OMICRON CMC 256 Plus.

Clamp-on CT probes were used to reduce the input values to comply with the equipment input limitations. The probes had two range settings as follows:

- Range 1: 0 - 10 A 100 mV/A
- Range 2: 0 - 80 A 10 mV/A

The first procedure to obtain energy data samples, is by connecting the OMICRON device at the main supply point in the switchgear room, at 9m level as illustrated in Figure 4.3 and 4.4. The difficulty of this method is that the cable length is measured at 80m between the measurement point and the machine room and has a 16mm^2 cross sectional surface area. Thus a voltage drop up to 3% is created across the installed power cable. The voltage drop offset must be taken into account with any energy calculations. The measured consumption is a summation of the elevator motor, controller, protection, car extraction fan and car lights. To obtain better results the different components should be extracted from the total consumption and analysed separately.

The second procedure is to take the measurements directly from the machine room equipment, where voltage drop can be neglected and additional measurement points are available. The software that was used to obtain and export the data samples was WINDAQ version 2.85. Equipment for this procedure included a DATAQ energy meter and 3 current probes with the following range settings:

- Range 1: 0 - 2 A 100 mV/A
- Range 2: 0 - 20 A 10 mV/A
- Range 3: 0 - 200 A 1 mV/A

The third procedure is a more permanent set-up, where an energy meter configuration was build and connected to the main elevator supply at the machine room for continuous data capturing and summation. The metering device measures the power signals and then calculates the power consumptions on a continuous basis. The energy consumption reading accumulates over time until it is reset. For the general energy measurements the M&V process started with the assembly of 5 energy measurement panels. The configuration details and the individual components were received from Asset Management Department within Eskom. The panels were then assembled and connected via CT's and VT's to each elevator's supply distribution board. The bill of material used in panel assembly is listed in Table 4.1 with the energy meter general layout in Figure 4.7 [7].

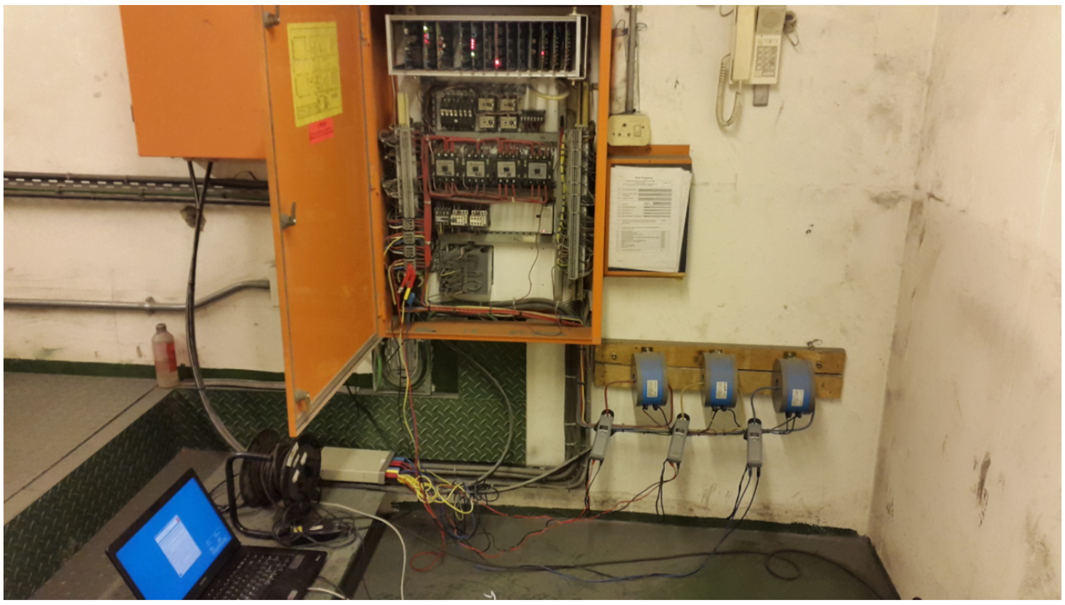


Figure 4.5: Actual Energy Measurements taken at Unit 1 Machine Room.

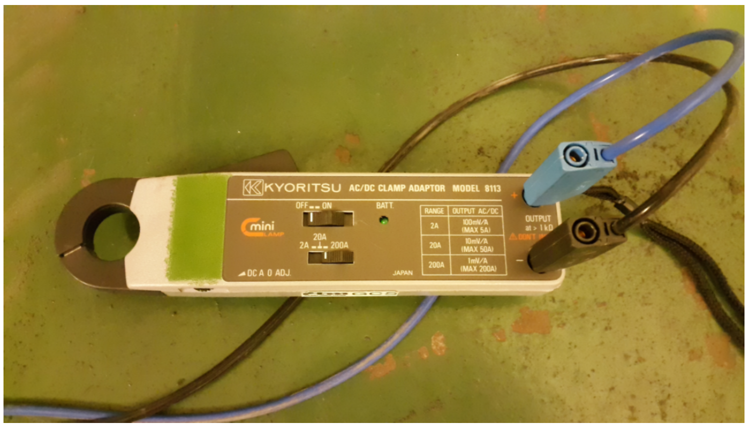


Figure 4.6: AC/DC Clamp Adaptor Used for Current Measurements.

There are two options to consider; either ring type or split-core type CT's and both are acceptable for measuring configurations. Ring type CT's can obtain higher VA ratings than split-core types and are more accurate and cheaper as well [6]. When using ring type CT's; the supply power must be isolated for the cables to be disconnected and the CT's to be inserted over the cables. According to the M&V guideline [6]: "CT's may never be open-circuited during live conditions, for a high voltage may be induced over the secondary terminals which can lead to electrical shock and/ or damage the CT." With split-core types the CT's can be installed over the cables, thus it can be done in live conditions. The accuracy of a CT is normally increased when its VA rating is limited [6], thus a 1A instead of a 5A Ring type CT was used for the purpose of this Thesis. Burdening should also be taken into account when choosing the

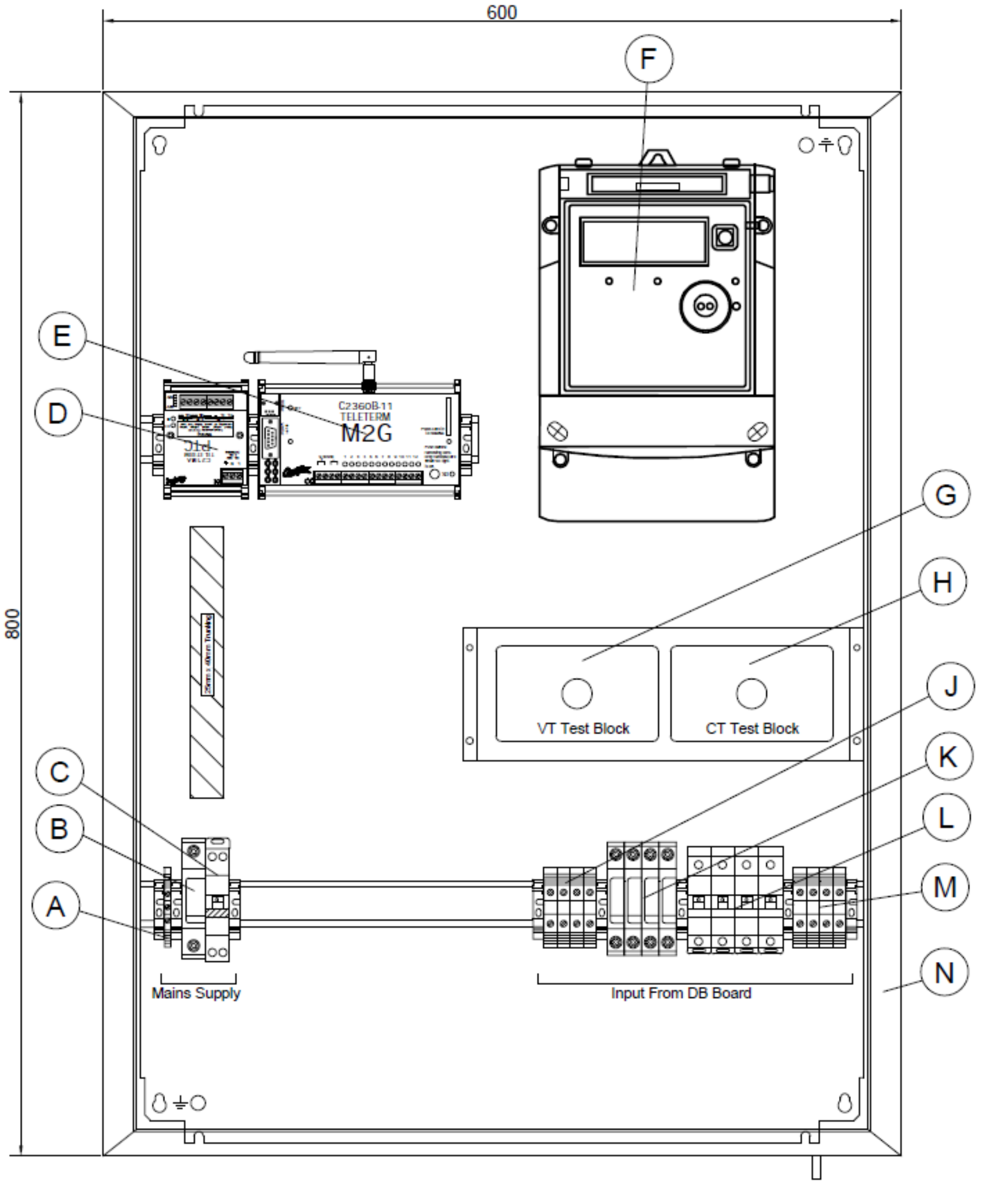


Figure 4.7: Measurement and Verification Energy Measurement Panel Layout [7].

correctly sized measuring cable. Burdening refers to the variations in supply voltages and currents, any electrical equipment is designed to handle. If the total burdening on CT's and VT's fall outside the designed burden the M&V data will be inaccurate. Voltage drop between the energy meter and the CT's and VT's must be less than the burden limitations on the equipment and is therefore calculated by the following equation [6]:

$$V_{\text{drop}} = \frac{V_{\text{Drop Value}} \cdot A \cdot L_{\text{cable}}}{1000} \qquad (4.3.1)$$

Table 4.1: Bill of Material of The Energy Measurement Panel [7].

ID	Description
A	Weidmuller Earth Terminal
B	Surge Arrestor
C	Hager MJN706 Double Pole Circuit Breaker
D	C2194A Poweterm PTC Power Supply
E	C2360b-11 Teleterm M2g
F	Landis + Gyr 3-Phase 4-Wire Meter
G	VT Test Block
H	CT Test Block
J	Spring Terminals
K	Surge Arrestors
L	Fuse Holders
M	Spring Terminals
N	Hager Siemens Grey (RAL7032) Mild Steel Enclosure 800x600x300
	Chassis Plate For FL124A
	Wall-Mount Brackets

Where the $V_{\text{Drop Value}} = 15.363\ mV/A/m$ for a 2.5mm^2 and 49.55 mV/A/m for a 4mm^2 cable, L_{cable} is the cable length in meters and A is the rated current. According to [6] the voltage drop should not be higher than the accuracy class index of the VT connected meter, which in this case is 0.5, thus a 2.5mm^2 cable will suffice for this application.

4.4 MOTOR DESCRIPTION

Table 4.2: Unit 1 Schindler Aux Bay 2-Speed Motor Nameplate Values

Description	18 pole conf.	4 pole conf.
Rated phase stator current	40 A	23 A
Rated line to line voltage	380 V	380 V
Rated stator frequency	50 Hz	50 Hz
Rated sync speed	333 rpm	1500 rpm
Rated power factor	0.83	0.83
Rated power	22 kW	10 kW

The motor utilised for the initial power consumption sampling and elevator modelling is from Unit 1 Schindler Aux Bay elevator at Majuba Power Station. The elevator motor is specified as a 3 phase AC synchronous motor with a 2 speed configuration. The windings are connected to an external controller

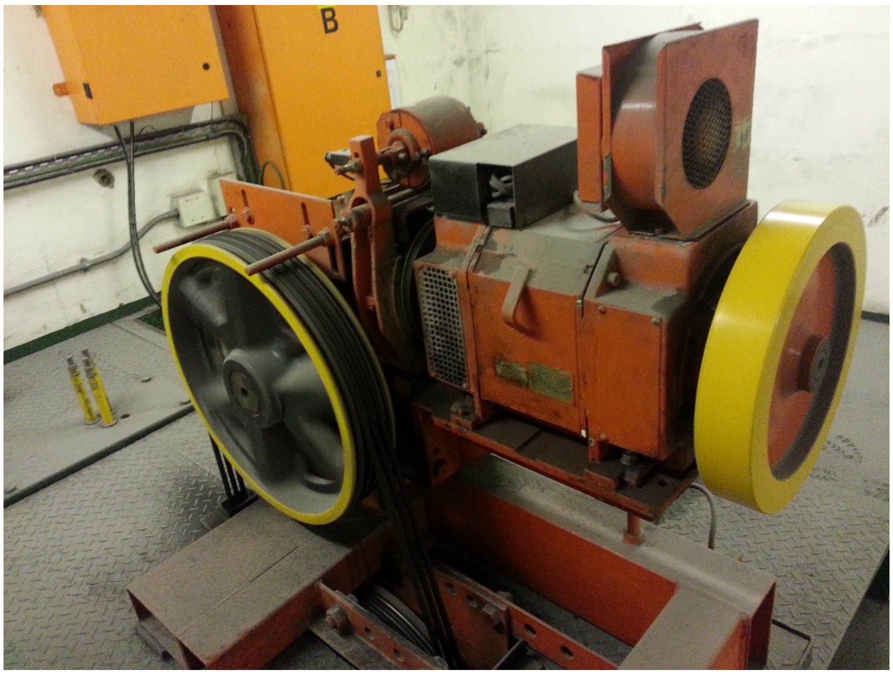


Figure 4.8: Schindler Aux Bay Elevator Drive Machine, Located at Majuba Power Station, Unit 1.

which selects between either 333 rpm or 1500 rpm synchronous speeds. Each speed setting has its own set of windings, where each is wound with 4 poles for high speed and 18 poles for the low speed setting. (Sync speed $= \frac{120 \cdot \text{frequency}}{\text{poles}}$). The motor has an open enclosure which allows for air cooling through vent openings. Just to note: this type of enclosure is not the most optimal enclosure to use for a power station which have a lot of contaminations in the air. The casing is also only rated for an ingress protection (IP) of 21, which only protect the motor against solid objects over 12.5mm and against liquids ingress to the degree of 1 which is vertically falling drops of water or condensation. Thus this motor is prune to overheating due to insufficient cooling and will cause insulation breakdown in the long run and increased running currents. Higher power losses will be occurred by this motor as a result of the additional heat loss occurred. In a typical coal powered power station you would expect an IP rating of 67 which is totally enclosed. The motor can have a non-ventilated enclosure where the heat is dissipated through the enclosure by means of conduction or by having an external fan which blows over the exterior of the motor to cool it down. The motor has a duty cycle of $S5$, as classified by the International Electrotechnical Commission (IEC), which refers to an“intermittent periodic duty with electric braking: Sequential, identical cycles of starting, running at constant load and running with no load and no rest period” [34].

The rated starting current ratio $\frac{I_A}{I_N}$ is given as 3.5 for direct on-line starting and with a 50% effective duty cycle (ED).

$$\%ED = \frac{\text{Braking time}}{\text{Total time for complete operating cycle}} \cdot 100 \tag{4.4.1}$$

With the total time for a complete operating cycle to be the summation of the acceleration time to reach set speed, run time at set speed, deceleration time to come to a complete stop and the time period the motor remains stopped.

Table 4.3: Theoretical Nameplate Motor Calculations.

Description	Symbol	Formula	4 pole conf.
Apparent 3 phase input power	S_{in}	$\sqrt{3}U_N I_N$	15138.12 VA
Active 3 phase input power	P_{in}	$S_{in} cos\psi$	12564.64 W
Reactive 3 phase power	Q_{in}	$\sqrt{{S_{in}}^2 - {P_{in}}^2}$	8443.49 VAR
Efficiency at rated speed	η	$\frac{P_{\text{out}}}{P_{in}}$	80 %
Stator angular speed	ω_{sync}	$\pi n_1 \frac{1}{30}$	157.08 rad.s^{-1}
Rotor angular speed	ω_m	$\pi n_r \frac{1}{30}$	150.8 rad.s^{-1}
Slip speed	ω_{slip}	$\omega_{\text{sync}} - \omega_m$	6.28 rad.s^{-1}
Slip	s_n	$\frac{\omega_{\text{slip}}}{\omega_{\text{sync}}}$	0.04 rad.s^{-1}
Torque at rated speed	T_e	$\frac{P_{\text{out}}}{\omega_m}$	66.34 Nm

Table 4.4: Unit 1 Aux Bay Elevator Actual Values and Measurements.

Symbol	Description	Value
V_N	Rated car speed	1 $m.s^{-1}$
V_C	Contract speed	2 $m.s^{-1}$
C_M	Suspension ratio	$2:1$
V_m	Rated motor speed	1440 rpm
ds	Sheave diameter	57 cm
r_g	Gearbox reduction ratio	43:2
m_{car}	No load car weight	1690 kg
m_{cw}	Counterweight mass	2200 kg
l_{rope}	Total Rope length	210 (3 X70) m
m_{rope}	Rope mass	119.7 kg
T_{rope}	Actual breaking load of rope	82 kN
m_{rated}	Rated full load car weight	2690 kg
I_{load}	Load inertia	315.9 to 397 $kg.m^2$
n	Overall system efficiency	0.49

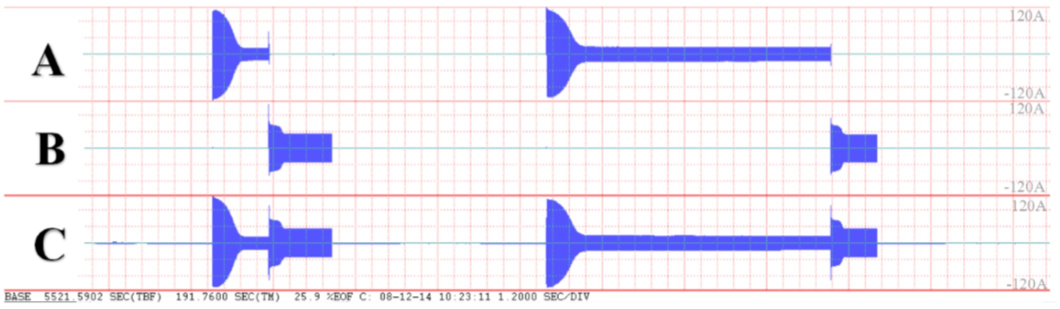


Figure 4.9: Two Speed Motor Current Sampling.

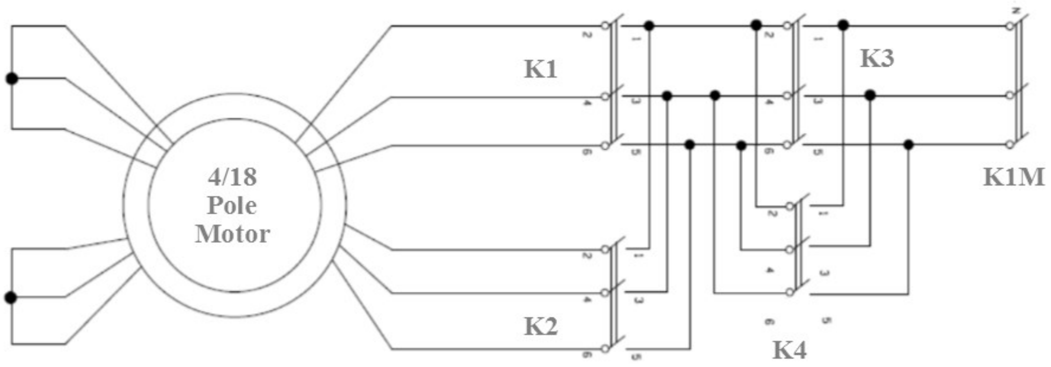


Figure 4.10: Aux Bay U1 Elevator Motor Terminal Schematic.

The elevator car rated speed is confirmed with Equation 4.2.14 and 4.2.15 with the sheave diameter and actual gearbox reduction ratio known. With reference to Table 4.4 it is estimated that this specific elevator configuration has an overall energy efficiency of about 49%. The objective of this Thesis is to improve overall system efficiency by improving elevator control techniques by reducing transportation cost for the same passenger throughput. However physical losses cannot be overcome by control philosophies, thus the elevator's mechanical and configuration needs to be upgraded as well to optimise energy efficiency. The system efficiency can be improved by upgrading the geared-traction configuration to the latest gearless machines with regenerative variable speed variable frequency drives (VVVF), which can reduce power consumption by 30% [35]. To establish the operation of the two speed elevator induction motor, measurements were taken at 3 different terminal points. With reference to Figure 4.9, waveform A refer to the current sampling taken from the 4 pole winding input terminals at $K1$, see Figure 4.10 . Waveform B refer to the 18 pole winding input terminals at $K2$ and waveform C refer to the main supply to the Elevator control cubicle, which includes all auxiliary power as well.

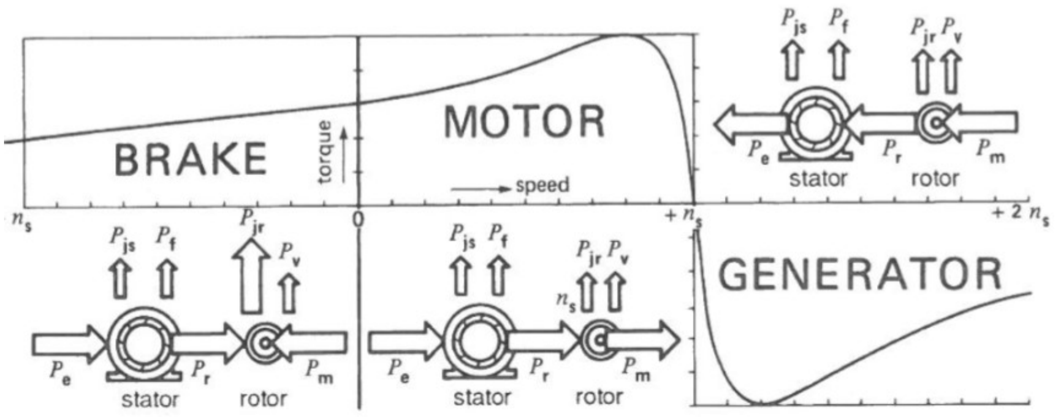


Figure 4.11: Complete Torque-Speed Curve of a 3-Phase Induction Motor [8].

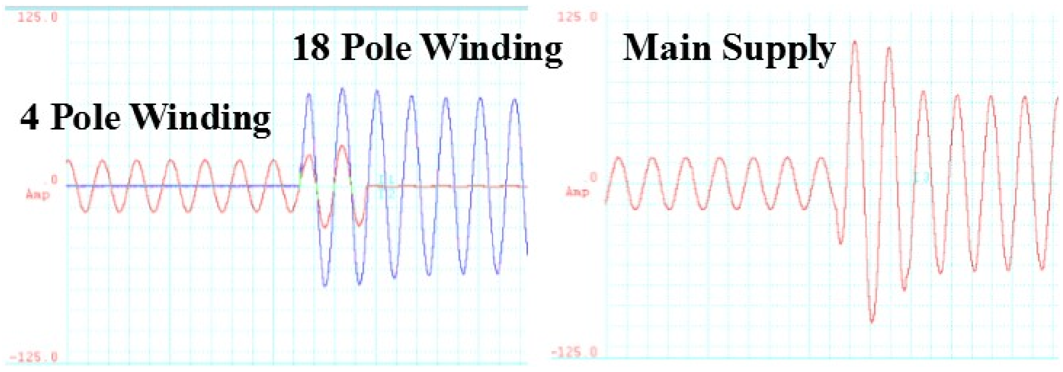


Figure 4.12: Two Speed Motor Winding Transitioning Period.

With reference to Figure 4.11 the two speed motor operates in different states, either in normal motor state at rated conditions, in a braking state or a generating state. When the motor is switched on the 4 pole windings are supplied by closing the 3 phase contactor ($K1$) to provide a closed circuit power. The initial direction of the motor, which results in the direction of the elevator car, is controlled via by the upstream contactors $K3$ and $K4$ by swapping the motor phases as required. The 4 pole winding configuration is responsible to accelerate the elevator car out of a stationary state into a rated speed state. As the elevator car approach its destination floor, the controller switches contactors $K1$ and $K2$ simultaneously. The transitioning period between the two configurations is shown in Figure 4.12 and lasts 42 milliseconds. The main peak supply current is measured at 101.9 A with the 4 pole winding at 29.2 A and the 18 pole starting current at 69.6 A.

During the transitioning period the motor goes into the braking state, when the 18 pole windings are energised. The phases are configured in the opposite direction to the 4 pole winding configuration, thus causing the motor to be at a slip of 5.3 ($\frac{333-(-1440)}{333}$) directly after switching. At this stage the rotor is effectively running backwards in the opposite direction to the field, which is known as plugging. The rotor will still receive electromagnetic power P_r from the stator and together with the mechanical power P_m, dissipated as heat [8]. During plugging the motor is prone to overheating and experience very high core losses which is not ideal from an energy efficiency perspective.

The reflective resistance $\frac{R_2}{s}$ from the single-phase equivalent circuit in Figure 4.13, represents the effect of the mechanical load (shaft load and rotor resistance) in relation to slip [9]. The power associated with $R2$ is the copper losses in the rotor, whilst the power associated with $\frac{R2(1-s)}{s}$ is the actual developed power, defined with [9],

$$P_m = P_{\text{gap}} - P_{\text{rotor}} = qI_2^2(\frac{R_2}{s}) - qI_2^2R_2 \ , \tag{4.4.2}$$

or equivalently

$$P_{\text{mech}} = qI_2^2R_2(\frac{1-s}{s}) \ , \tag{4.4.3}$$

where q is the number of stator phases. The electromechanical power P_{mech} is therefore positive with $0 < s < 1$ and negative with $s < 0$ and $s > 1$. However during plugging the torque will remain positive, because of higher electromagnetic power; resulting in slowing down the rotor slip from 5.3 to just above zero.

If the direction of the rotating field were not reversed during the transitioning period, the new slip would be -3.505 ($\frac{333-1440}{333}$). By applying Equation 4.4.2 and 4.4.3 with a negative slip, the motor will have a surplus of electromechanical power with fewer core losses. Thus the machine would have been operating in the generating state and performed regenerative braking, where a negative torque decelerates the motor, with the surplus kinetic energy reverted back through the regenerative drive until the motor comes to a stationary position. The latest elevator applications uses this potential regenerative capabilities to put power back into the system and to offset the elevator's auxiliaries.

After plugging the motor will attempt to go into the normal motor state and accelerate to its second rated speed in the same direction of the field. However before the actual rotor direction can be changed the supply to the motor is removed by closing the motor supply contactor, thus stopping the elevator car at this point. This means that the 18 pole winding configuration were never intended to provide a second speed for the elevator car, but rather utilised as a braking mechanism for the elevator application. The additional pole pairs in the same air gap as the 4 pole configuration, will effectively increase the required magnetising current in the core and will lead to a lower power factor at rated speed. During the transitioning period; the amount of flux per pole will be decreased because the total air gap flux will now be divided by

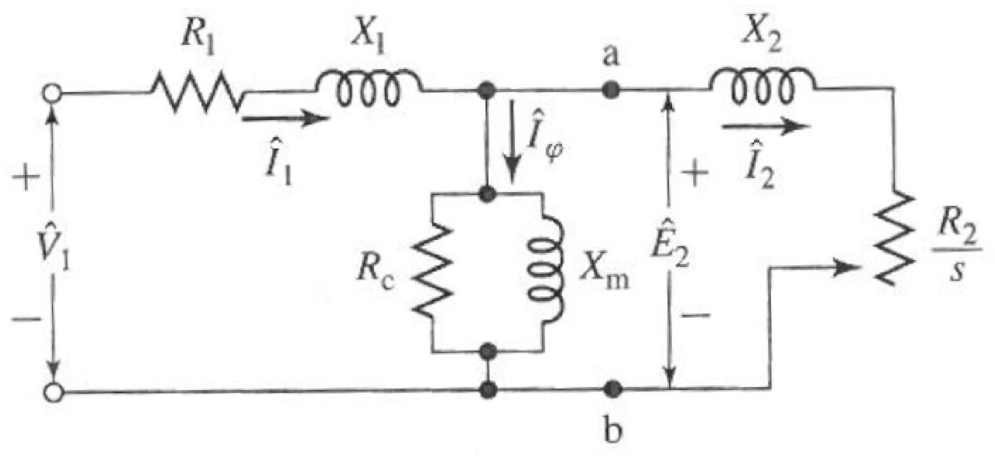


Figure 4.13: Single-Phase Equivalent Circuit For a Polyphase Induction Motor [9].

9 pole pairs instead of only 2. A correctly sized 18 pole motor will deliver much higher torque for the same output power with comparison to a 4 pole motor, due to the lower turning speed. However the physical size of this motor is less than you would normally expect with this number of pole pairs, thus it can be assumed that the number of winding turns per pole pair as well as the amount of active iron mass are less than required to bring this machine to rated speed. This will result in a weakened air gap field and lower generated torque capabilities.

There are another scenario in the elevator application where the motor can enter the generating state other than bringing the elevator car to a stationary position. When an unbalanced load is in the same direction as the elevator car; the synchronous speed is exceeded, thus creating negative electromechanical power with $s < 0$. Thus a negative torque will be applied at this point to avoid an over-speeding condition and to bring the elevator car back to rated speed.

4.4.1 GRAPHICAL REPRESENTATIONS AND TRENDING

The actual power consumption of the elevator application has been established by the M&V process as described in this chapter. The format of the actual samples is graphically illustrated by Figure 4.14 together with Table 4.5. The sample length is with relation to the trip information provided by Table 8.3 and 8.4 as an example. The same can be done for all valid floor landing permutations in the up direction as well as the down direction. In the specific case of the Aux Bay U1 elevator; we have 12 different floor combinations between

0m, 9m, 16m and 20m landings. The valid floor combinations are then further extended to include load percentages, which are divided into fractions of 10 percentage points of the maximum load capacity. Thus we have 132 sample points to consider.

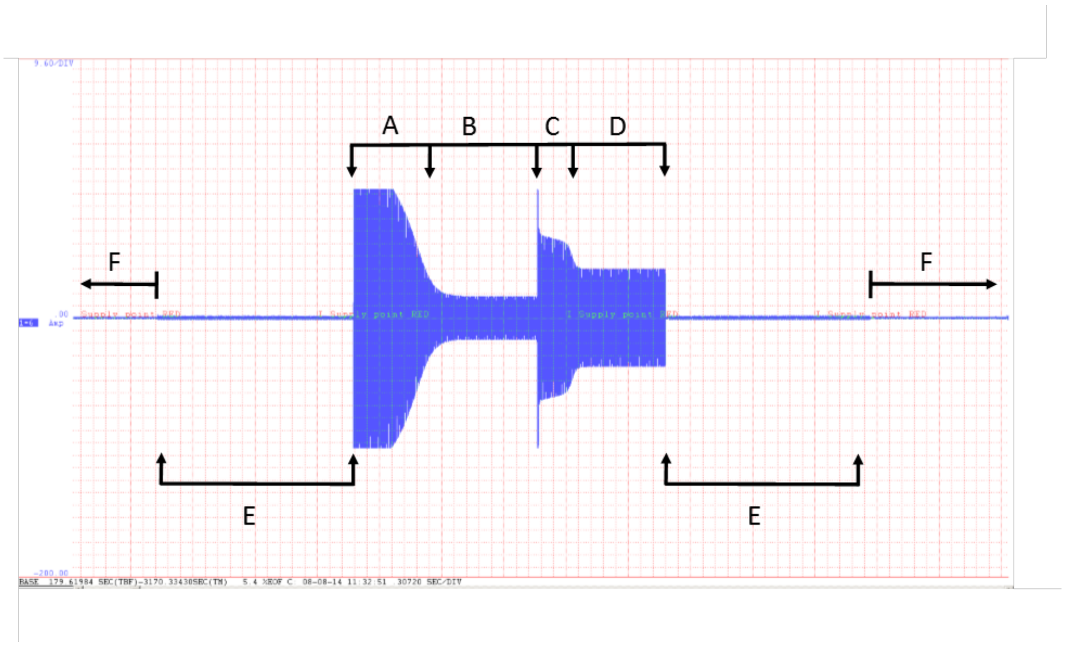


Figure 4.14: Main Supply Current Sample for a 20m to 16m Elevator Cycle Period.

Table 4.5: Current Sample Descriptions, Refer to Figure 4.14.

Symbol	Description
A	Acceleration
B	Rated Speed
C	Deceleration
D	Reduced Speed
E	Elevator Doors opening and closing
F	Standby consumptions

When actual samples have to be recorded it is not always feasible to obtain all permutations, for it can be time consuming and sometimes difficult to conduct during busy working environments. For 10 floors the number of required samples already reach 900. With various machine learning techniques, the required samples can be severely reduced to only a few representative samples. For the

Table 4.6: Actual Measurements for 8 Persons Going Down From 20m to 16m , Refer to Figure 4.14.

	Current (A_{rms})	Duration (sec)	Voltage (V_{rms})	Power (W)	Energy (J)
Floor height	20 - 16	20 - 16	20 - 16	20 - 16	20 - 16
Load	8 People	8 People	8 People	8 People	8 People
Acceleration period (A)	62.54	1.73	222	34,162	74,894
Rated speed period (B)	11.56	2.60	226	8,297	20,669
Motor 4 Pole period (A + B)	40.60	4.33	225	24,367	113,951
Deceleration period (C)	43.36	1.06	224	23,175	21,190
Reduced Speed period (D)	28.10	1.62	225	14,846	35,754
Motor 18 Pole period (C + D)	34.81	2.69	225	17,525	58,230
Full duration (A,B,C,D)	38.47	7.01	225	21,772	174,080
Doors closing (E)	3.08	4.85	226	1,498	7,091
Doors opening (E)	1.02	4.85	226	751	3,741

Table 4.7: Actual Measurements for 1 Person Going Down From 20m to 16m , Refer to Figure 4.14.

	Current (A_{rms})	Duration (sec)	Voltage (V_{rms})	Power (W)	Energy (J)
Floor height	20 - 16	20 - 16	20 - 16	20 - 16	20 - 16
Load	1 Person	1 Person	1 Person	1 Person	1 Person
Acceleration period (A)	61.80	2.19	222	34,140	74,847
Rated speed period (B)	14.73	2.49	225	8,266	20,592
Motor 4 Pole period (A + B)	43.59	4.68	224	24,280	113,540
Deceleration period (C)	41.47	0.91	224	23,166	21,181
Reduced Speed period (D)	26.48	2.41	225	14,829	35,713
Motor 18 Pole period (C + D)	31.30	3.32	225	17,517	58,205
Full duration (A,B,C,D)	38.93	8.00	224	21,724	173,693
Doors closing (E)	2.66	4.73	226	1,497	7,088
Doors opening (E)	1.33	4.98	226	750	3,738

purpose of this case study, the actual samples have been utilised with conjunction with a created Neural Network to establish true trending and accurate distribution patterns across the different combinations and load percentages. The developed Neural Network Energy Model is described in Section 2.3.5 and Chapter 7.

With reference to Figure 4.15 the two speed motor supply currents are compared to the distance travelled and the load imbalances of the elevator car, counter weight and the subjected loads. It is clear that the distance travelled don't have an influence on the 18 pole configuration supply current, which is used to decelerate and stop the elevator car before the destination floor is reached. However when the elevator car is empty and traveling in the up direction; the highest current is noticed, as well as when it is at capacity traveling downwards. This load imbalance is creating a higher moment of inertia in the same travelling direction, thus higher current is required to overcome the higher inertia to decelerate, as opposed to when the load imbalance is in the opposite direction of travel. In contrast to this, the higher inertia reduces the amount of generated torque required to reach the rated speed by the 4 pole configuration, and increases the required input power when the load imbalance is working against the law of gravity. High rms supply currents are experienced during short distance trips, irrespective of the direction, because when the motor is accelerated high starting currents are expected, but just as the motor is reaching rated speed the destination is almost reached and the car is stopped. Thus the total rms supply current is lower at longer distances where the time spend at rated speed reduces the overall current and input power average for a trip, see Figure 4.16.

The elevator car running time is increased with distance travelled as expected. However the time gradient with relation to distance is not a 1^{st} order linear model as seen in Figure 4.17. The order of complexity is increased by the acceleration and deceleration of the motor with varying loads and distances. Cogging torque is also experienced during the transitioning period of the different winding switching, thus estimating exact running times becomes very complex. Also to be noticed is that if the elevator car is traveling downwards and is heavier than the counter weight the running duration is reduced as a result of supporting gravitational forces. With increased transported loads; the overall elevator configuration weight also increases, which results in higher inertia as theoretically calculated in Section 4.2.2, with Equation 4.2.8.

4.5 SUMMARY AND CONCLUSIONS

The elevator power consumption philosophy was introduced in this chapter. The general philosophy is to create an energy model for our elevator configuration based on theoretical calculations and actual energy samples. The theoretical model included references to Newton's 2^{nd} Law for the applied forces to the sheave, which resulted in a nett torque. The overall inertia at the sheave was given as the sum of the load inertia and that of the sheave and the shaft. Further the influence of the gearbox was investigated and factored into the overall system efficiency. Kinetic and potential energy also made part of the model where the elevator speed, suspension ratio and gearbox reduction ratio were taken into account. The Measurement and Verification (M&V) concept was introduced where different procedures to obtain the actual energy samples were conducted and compared to each other. The best procedure was proven to be the method where the measurements are directly taken from the machine room equipment, where voltage drop can be neglected and additional measurement points are available. The actual energy samples was then compared to the theoretical calculations based on the rated nameplate values, as well as the developed equations from this chapter. The overall system efficiency was concluded to be around 49%. The exact operations of the two speed motor were discussed and directly concluded from the graphical representations and compared to the equivalent single phase circuit equations.

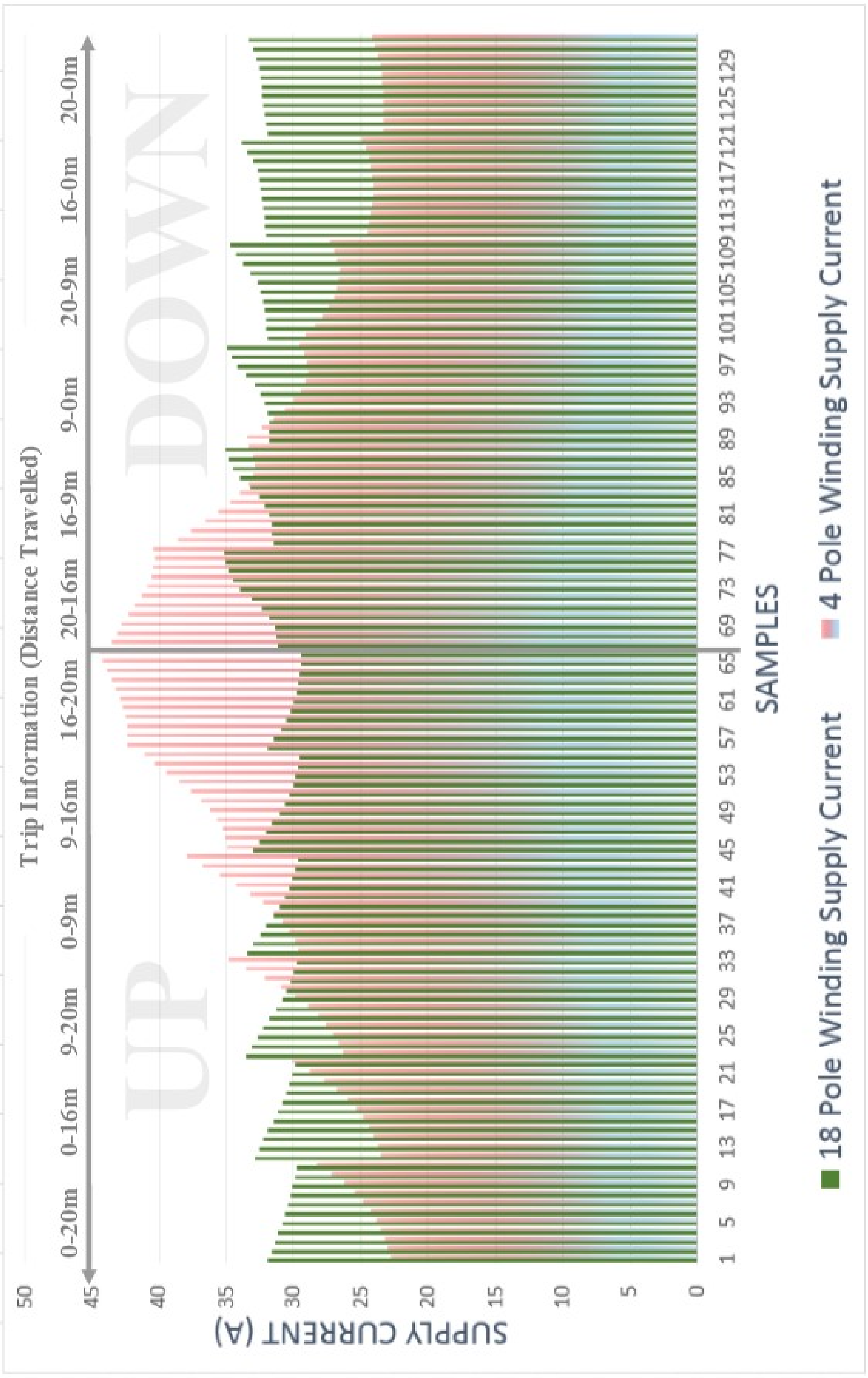


Figure 4.15: 18 Pole and 4 Pole Supply Current (rms) vs. Distance Travelled and Load Percentages (0-100% of rated load per trip distance).

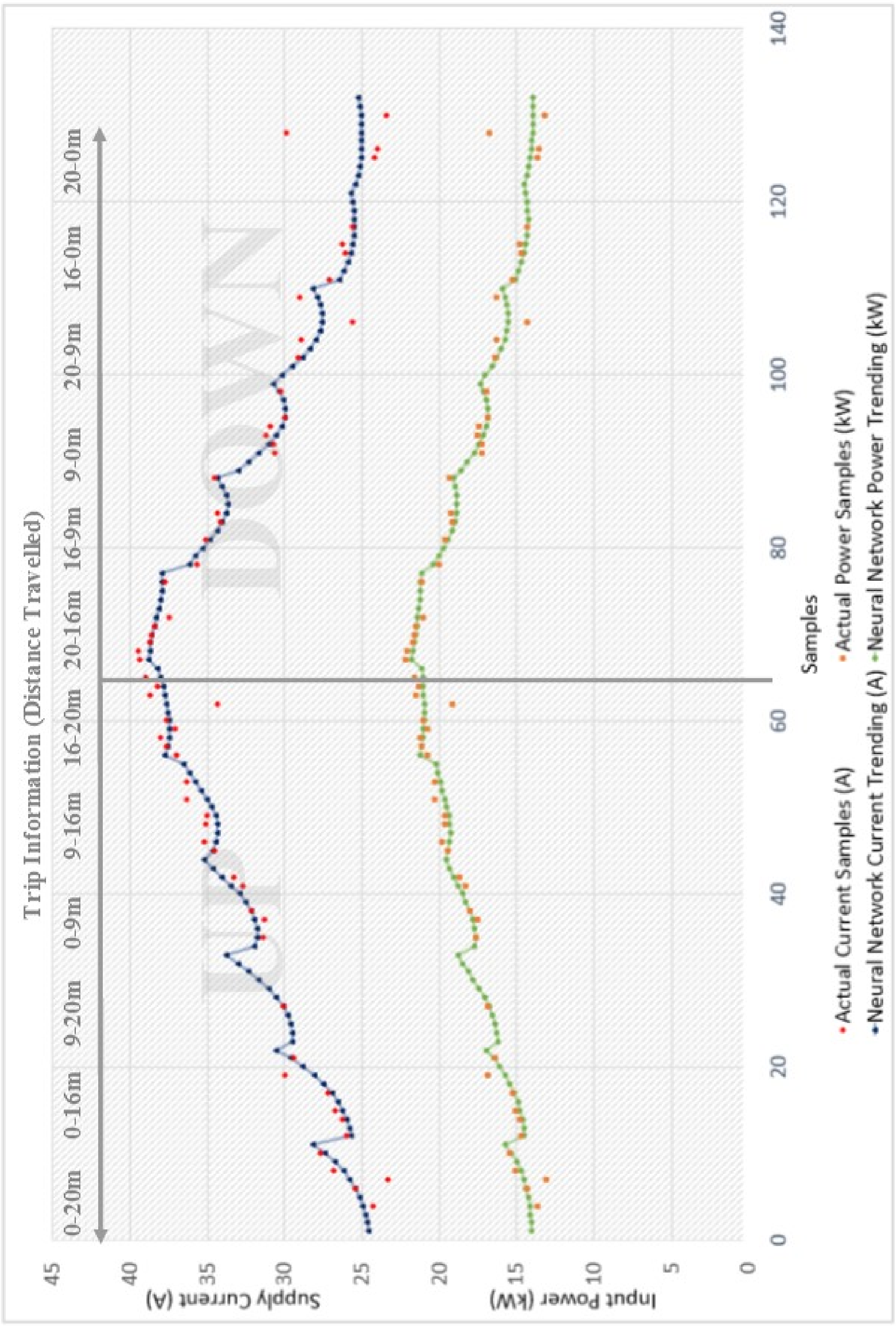


Figure 4.16: Elevator Supply Current (rms) and Input Power vs. Distance Travelled and Load Percentages (0-100% of rated load per trip distance).

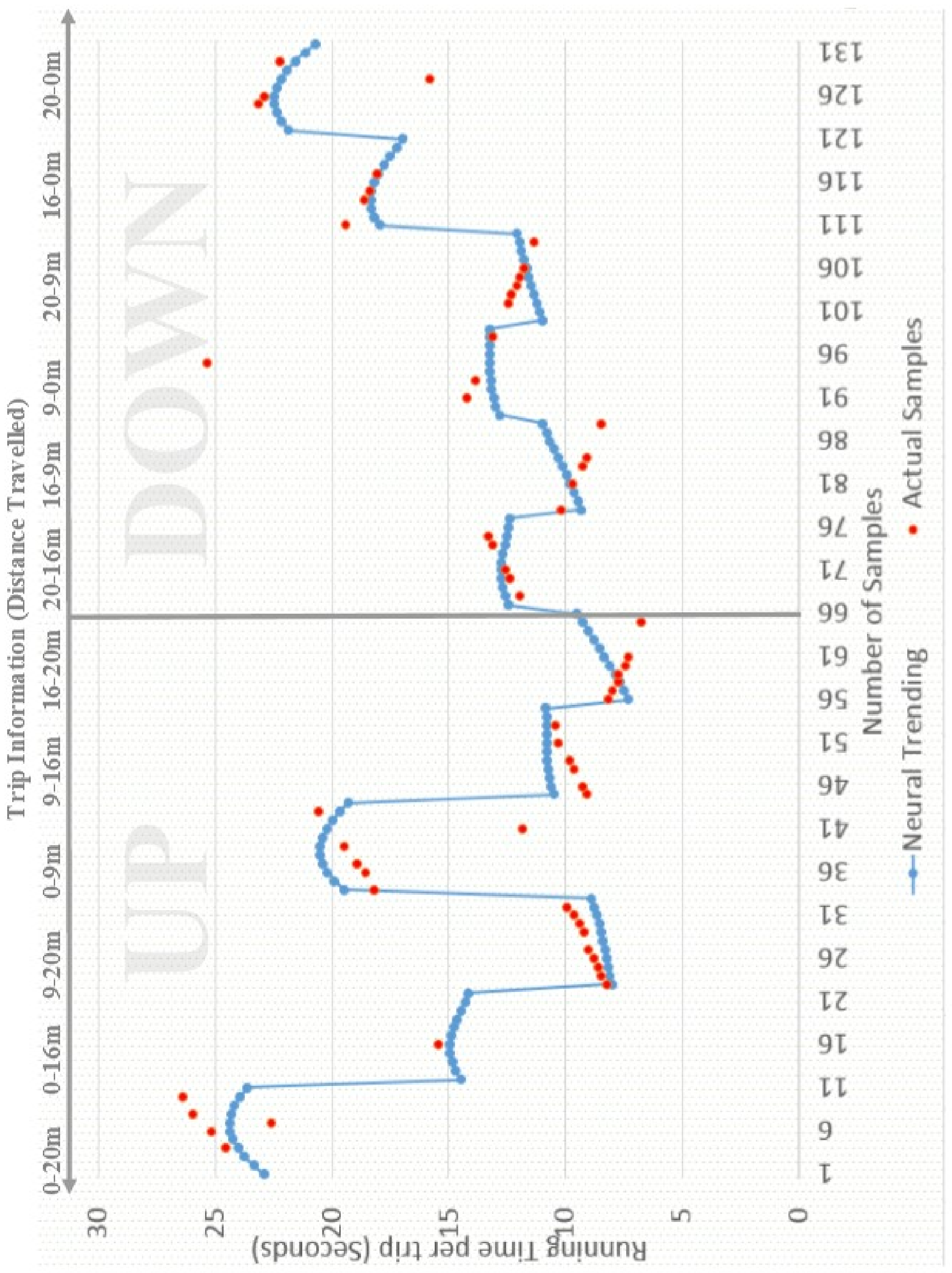


Figure 4.17: Elevator Running Time vs. Distance Travelled and Load Percentages (0-100% of rated load per trip distance).

It was realised that each approach to create an accurate energy model had a major shortcoming that needed to be addressed. With any theoretical model, there are always uncertainties and unknown variables, which are most often neglected and ignored. Thus exact equations becomes very difficult to obtain and to solve and the shortcoming with energy sampling, is that it is not always feasible to obtain all permutations, for it can be very time consuming and sometimes difficult to conduct during busy working environments. For example for a 10 storey building, the number of required samples reaches 900 for all known sample permutations. A solution to the identified shortcomings can be found with various machine learning techniques, where the required samples can be severely reduced to only a few representative samples. The actual samples have been utilised in conjunction with a created Neural Network to establish true trending and accurate distribution patterns across the different combinations and load percentages. The developed Neural Network Energy Model is described in Section 2.3.5 and Chapter 7. With the implementation of Neural Networks, the unknown variables can be accurately modelled without the need to define them.

The created energy model is then ultimately used to estimate the energy consumption for any floor order route and for accurately predicting the actual running times of an elevator car, without making any assumptions, as it would have been the case with a theoretical model. The energy model can then be incorporated in the TSP and the VRP later on in this Thesis, where the shortest route is calculated between the possible floor orders.

Chapter 5

ELEVATOR PROBABILITY THEORY

5.1 PROBABILITY PHILOSOPHY

The concept of probability can be incorporated throughout the development of the AI controller. Any aspect of the decision making process, from the input data to the optimal developed routes can be predicted to a certain extend. The difference between a normal elevator controller and the AI controller, we are developing, is that we are not only looking at present input data, but are also interested in past and future information. We are to process previous collected traffic data and use it to estimate future traffic occurrences. The controller can benefit from any additional information about future events and can use this information to calculate possible routes or to dispatch a elevator proactively in the direction of the expected traffic. With probability factored in and incorporated with the present available data, the elevator controller can prioritise better between floors and make more accurate decisions with the available information.

After a fully functional probability model has been developed, it can be utilised to create pre-planned elevator car responses to the probability of future passenger requests. The correlated responses to the predicted traffic pattern results in pre-planned routes. The pre-planned routes can also be optimally utilised to establish the starting positions for the elevator cars or to which floor the various elevator cars should be parked when going into standby-mode. Optimal resting and starting positions are important to reduce the cost between the initial position and the floor which will be serviced next. Another use for priori routes is to estimate the expected cost for a time period, either expected transportation cost, number of trips or overall expected power consumption.

By predicting future traffic patterns we are also able to estimate when the elevator will be in its least occupied time period and the duration, where it will be most optimal to do routine maintenance on the specific elevator. As well as establishing or recommending a time period when some elevators can be switched off completely, because of inactivity and the lack of future necessities, with at least one elevator still kept on standby.

However with the dynamic nature of an elevator application, where traffic flow is continuously changing, pre-planned routes can quickly differ from the optimal path to follow. Thus it is important to have error feedback and correction between the planned and optimal routes as required.

5.2 FLOOR ORDER PROBABILITY

5.2.1 PTSP INTRODUCTION

Priori or pre-planned routes are defined as routes that specifies an ordering of all the possible floors which need to be serviced for a time period or per call instance. These routes create a proposed schedule for a time period and can be used for regularity purposes. The floors with neglectable passenger demand probabilities do not need to be included in the planned route and can be excluded from the different algorithms to potentially reduce overall computational power requirements.

The Probabilistic Traveling Salesman Problem (PTSP) is similar to the classical TSP as described earlier but with service probabilities, assuming independence between floors. The PTSP is attempting to deal with the uncertainty of routing problems, like nondeterministic cost calculations and uncertainty in passenger demand at each floor.

The Probabilistic Traveling Salesman Problem with Deadlines (PTSPD) can be further classified and implemented when additional time window constraints are added to the problem. Jaillet [36] illustrates through his research that given you have an optimal tour A for the general TSP and an alternative tour B. The total length of tour A is calculated to be shorter than that of B with the condition that every city is to be visited in the planned route. He then goes on to proof that if one or more of the cities are not going to be visited for whatever reason, there exists a more optimal route that should have been followed. For instance let's give each floor in route A and route B a probability of 0.5 (chance to be visited), then it can actually come to past that the length of tour A is 30% more than that of route B. This same notion proved by Jaillet [36] can be applied in terms of an elevator PTSP application, to generate the optimal floor order with the probability of some floors not being visited taken into account.

We can compute the probabilities of not visiting a number of floors in the building for a given sequence. For a building with ever-changing passenger demands throughout the day, it becomes important to take the service probabilities into account when generating the planned routes for every elevator car in the elevator group. For instance the event A, can be described as a floor that are not visited in a planned route, and can be defined with Bernoulli trail probability,

$$P\{\text{event A: only k number of occurrences} = (\begin{matrix} n \\ p \end{matrix})p^k(1-p)^{N-k} \ , \tag{5.2.1}$$

with k the number of floors being skipped in the planned route with length N and probability to skip a floor are set to be equal and given as p. The probability to skip a floor is also statistically independent for the planned route. For example let's say we have 8 floors and a planned route length of 8, the probability to skip a floor is estimated at 0.4, thus: $P(0) = 0.017$, $P(1) = 0.09$, $P(2) = 0.21$, $P(3) = 0.28$, $P(4) = 0.23$, $P(5) = 0.12$, $P(6) = 0.04$, $P(7) = 0.01$, $P(8) = 0,0007$. The probability to skip for instance 3 or less floors in a planned route is given by $P(0) + P(1) + P(2) + P(3)$ to be almost 60 %.

5.2.2 PTSP COMPUTATIONS

The probabilistic traveling salesman problem (PTSP) is defined as finding the optimal floor order which will reduce the overall tour cost with a set of floors $N = i|1, \ldots, n$ with respective service probabilities $P = p_i|1, \ldots, n$ [28]. Bayes' theorem can be used together with the priori floor probabilities, $P(F_{X,Y})$ to establish the probability of servicing the next floor, as illustrated in Figure 5.1

The Bayesian theorem connects the respective prior probabilities with the posterior probabilities as follows:

$$P(B_i|A) = \frac{(P(A|B_i)P(B_i)}{P(A)} \tag{5.2.2}$$

$$P(A) = P(A|B_1)P(B_1) + \ldots + P(A|B_n)P(B_n) \tag{5.2.3}$$

Each step in the floor order sequence is written in the format $F_{X,Y}$, where the X donates the position in the sequence and Y the respective floor number. We have $X = 0, \ldots, k$, with k the total number of stops in the planned floor order sequence and $Y = 0, \ldots, N$, with N the total amount of floors. The respective

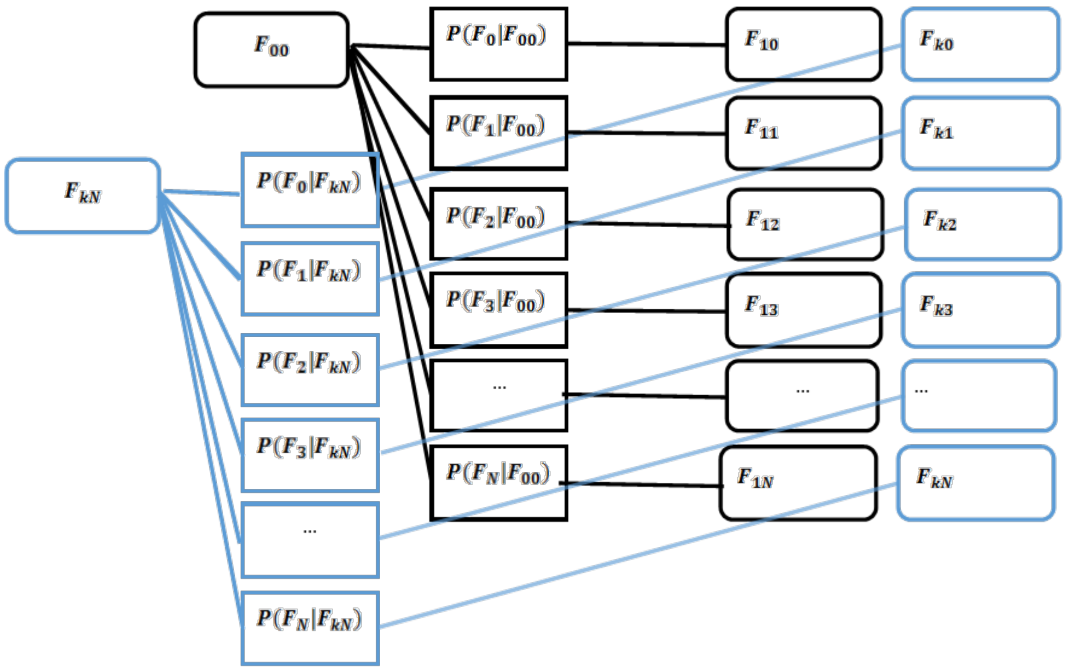


Figure 5.1: Floor Order Probability Computations

transition probabilities are given as $(P(F_Y|F_{X,Y})$ and the posteriori probabilities are $P(F_{X,Y}|F_Y)$. For example in order to calculate the probabilities for the next floor in the sequence to be serviced, for instance the 2nd position in the sequence, we have $k = 1$ and require prior probabilities: $P(F_{1,0})$ to $P(F_{1,N})$. If the 1st position in the sequence was for example known to be floor 3 beforehand, the probabilities required are then: $P(F_{1,0}|F_{0,3})$ to $P(F_{1,N}|F_{0,3})$.

5.2.3 FLOOR ROUTING AMBIGUITY

Wherever there is possibility of ambiguity in the proposed floor order, fuzzy logic theory can be used to solve these problems. Ambiguity can arise when there are maybe too many choices; for instance there are more than one route that provides the same optimal calculated cost. With fuzzy logic the best route can be chosen from one of the optimally perceived routes. Ambiguity can also arise when there is a contradiction between decisions being made or between available data that is in contrast with the prevailing occurrences. When classification problems are defined or analysed, often classes can overlap when the features vector $X = (x_1, x_2, \cdots, x_m)$, extend over more than one class. One way to separate samples into classes is according to the probabilities of the ambiguous classes. One method to choose a class c_i is by choosing the maximum posterior probability, $P(c_i|x) \geq P(c_j|x)$ from general Bayesian theory, see Equation 5.2.2 and 5.2.3. The decision function is then $r_i(x) = P(c_i|x)$ in this case and is proven to provide the minimum classification error [27].

5.3 EXPECTED REQUESTS

5.3.1 INTRODUCTION

The assumption were made that at any moment in time the controller know how many people are waiting, at which floor, how long and also each passenger's destination floor. What the controller doesn't know is any information about future requests. A request, r is defined as, (τ_r, s_r, d_r), where τ_r is the time the request is received, s_r is the starting floor and d_r the destination floor. Let's make the assumption that the controller know the number of people currently present in the building and at which floor level, then we can define a few population variables as follows:

- O_i represents the total population at floor i and total building population, O, which are not necessarily in the building,
- U_i represent the number of people that are currently on floor i and U, the total actual population in the building,
- K_i can now present the population that have not reached floor i and K the total population that are not in the building or have not reached their floors, with $K_i = O_i - U_i$.

5.3.2 POISSON PROBABILITY

The probability to receive a request $P(X)$ can be obtained by knowing the respective Poisson arrival probability functions for each floor. Let's assume a passenger arrival rate of λ_i at each floor i, with $i = 1, \cdots, N$ and N the number of floors. λ_i is defined as the average number of passenger arrivals per floor in a specific period T:

$$b = \lambda T \tag{5.3.1}$$

The general Poisson probability density $f(x)$ and the Poisson probability distribution $F(x)$ are defined as follows for $b > 0$ [37]:

$$f(x) = \mathrm{e}^{-b} \sum_{k=0}^{\infty} \frac{b^k}{k!} \delta(x - k) \tag{5.3.2}$$

$$F(x) = \mathrm{e}^{-b} \sum_{k=0}^{\infty} \frac{b^k}{k!} \mu(x - k) \tag{5.3.3}$$

With the mean and variance defined as:

$$\bar{X} = b \tag{5.3.4}$$

$${\sigma_x}^2 = b \tag{5.3.5}$$

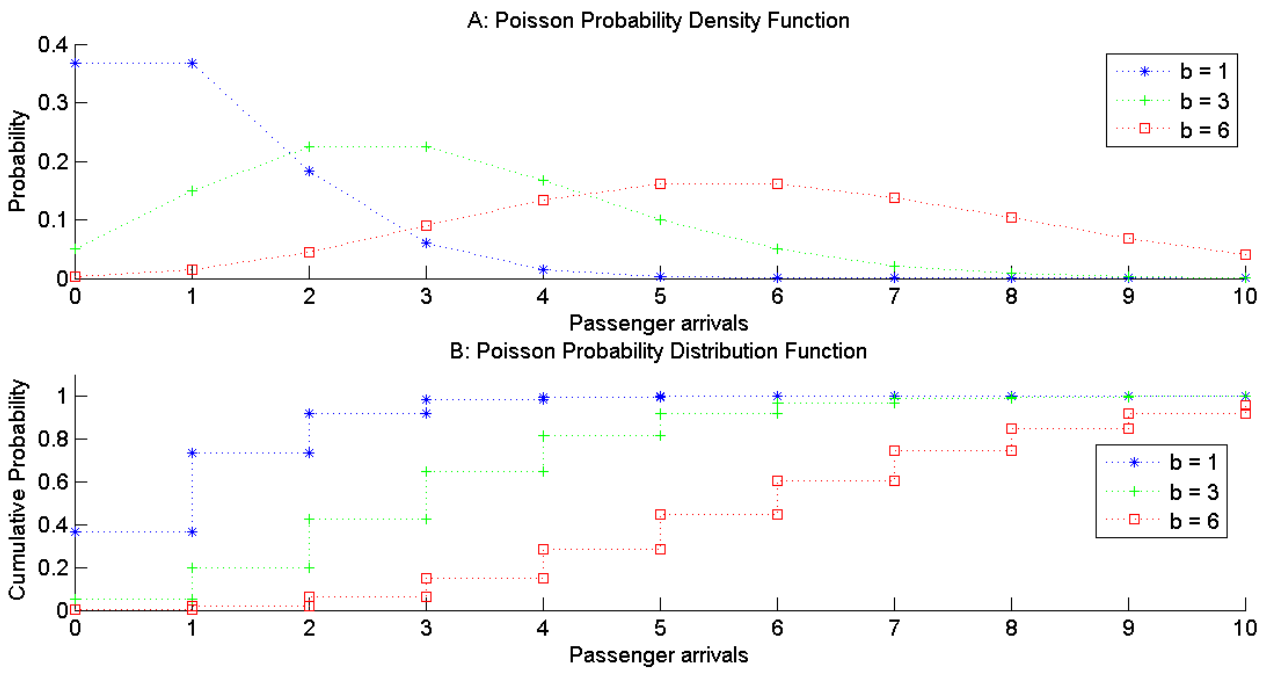


Figure 5.2: Poisson Probability Functions.

The above probability functions are used to provide a distribution model for stochastic passenger arrivals at each floor. Which means that the probability for a specific time is zero, however as a result of a repeating phenomenon called the Poisson process we are able to sufficiently describe the arrival patterns. The first important property to notice is that the Poisson counting process $K(t);\ t > 0$ has stationary increments, which means that the distribution of passenger arrivals only depends on the length of the interval, t and not on the starting point t'. This means that $F_{N(t'-t)}(x) = F_{N(t')}(x) - F_{N(t)}(x)$ for every $t' > t$ [38]. The second property to notice is that all Poisson increments are independent from each other, which means that each passenger arrival has no influence on the probability of another passenger's arrival.

5.3.3 DIRECTIONAL PROBABILITIES

Through the Poisson process it is established that $K_i(x)$ passengers are to be arriving at a rate of λ_i at floor i during an interval T. To provide more information in regard to the possible traveling direction; it can be established that each passenger would like to go up with a probability of ρ_i and down with a probability of $1 - \rho_i$.

The Poisson process has now been subdivided into two independent Poisson processes with arrival rates of:

$$\lambda_{i\_\text{up}} = \rho_i \lambda_i \tag{5.3.6}$$

$$\lambda_{i\_\text{down}} = (1-\rho_i)\lambda_i \tag{5.3.7}$$

The number of expected destinations from floor i with the number of passenger arrivals; $K_i(x) = K_{i\_\text{up}}(x) + K_{i\_\text{down}}(x)$, can be estimated as follows:

If the expected destinations are equally likely and uniform with $\frac{1}{N_i}$ for every passenger, with N_i the possible number of destination floors. This means that there are $N_i = N_{i\_\text{up}} + N_{i\_\text{down}}$ possible destinations in the two directions from floor i. The probability that none of the $K_{i\_\text{up}}(x)$ passengers will be going up to the same floor is given by:

$$(1 - \frac{1}{N_{i\_\text{up}}})^{K_{i\_\text{up}}(x)} \tag{5.3.8}$$

The probability that at least one person from the $K_{i\_\text{up}}(x)$ passengers will be going up to a specific floor is then:

$$1 - (\frac{N_{i\_\text{up}} - 1}{N_{i\_\text{up}}})^{K_{i\_\text{up}}(x)} \tag{5.3.9}$$

The expected number of different destinations in the up direction from floor, i can now be estimated as:

$$E[D_{i\_\text{up}}] = N_{i\_\text{up}}[1 - (\frac{N_{i\_\text{up}} - 1}{N_{i\_\text{up}}})^{K_{i\_\text{up}}(x)}] \tag{5.3.10}$$

In the same way can the expected number of different destinations in the down direction from floor, i can be defined as:

$$E[D_{i\_\text{down}}] = N_{i\_\text{down}}[1 - (\frac{N_{i\_\text{down}} - 1}{N_{i\_\text{down}}})^{K_{i\_\text{down}}(x)}] \tag{5.3.11}$$

The expected number of different destinations from floor i are now: $E[D_i] = E[D_{i\_\text{down}}] + E[D_{i\_\text{up}}]$. From the planned route it is already known how many stops each elevator car will make from each floor i depending on how many floors still to be visited in either direction. Thus together with the expected number of destinations from future requests; the number of additional stops have now been established.

5.3.4 EXPECTED DISTANCE OF TRAVEL

The effects of future requests on any planned route are two-fold. The first is that the elevator car is expected to make potentially more stops in order to deliver the additional passengers to their destinations and the second is to travel further up or down, before the direction of travel is reversed to continue with the return route. The expected number of additional stops were discussed in Section 5.3.3 and the expected additional distance to travel shall now be provided.

When the rate of incoming requests has been established, the controller needs to prepare how to deal with the future requests and to minimise their effect on the already planned routes. The highest reversal floor H can be estimated from any floor i as follows:

By using Equation 5.3.8, the probability that any of the passengers $P = K_{i\_\mathrm{up}}(x)$ will not travel higher than the j^{th} floor is given by:

$$(1-\frac{1}{N_{i\_\mathrm{up}}})^{P}\cdot(1-\frac{1}{N_{i\_\mathrm{up}}-1})^{P}\cdot\ \ldots\ \cdot(1-\frac{1}{j+1})^{P} \tag{5.3.12}$$

with $\frac{1}{N_{i\_\mathrm{up}}}$, the probability that one passenger will go to floor j, and N_{i_up} the total number of floors above the current floor, thus Equation 5.3.12 is further reduced to:

$$(\frac{j}{N_{i\_\mathrm{up}}})^{P} \tag{5.3.13}$$

The probability that i is the highest floor is given by:

$$(\frac{j}{N_{i\_\mathrm{up}}})^{P}-(\frac{j-1}{N_{i\_\mathrm{up}}})^{P} \tag{5.3.14}$$

The average highest reversal floor H is then [1]:

$$H = N_{i\_\mathrm{up}} - \sum_{j=1}^{N_{i\_\mathrm{up}}-1}(\frac{j}{N_{i\_\mathrm{up}}})^{P} \tag{5.3.15}$$

with unequal floor populations or probabilities the average highest reversal floor H is defined as [1]:

$$H = N_{i\_\mathrm{up}} - \sum_{j=1}^{N_{i\_\mathrm{up}}-1}(\sum_{i=1}^{j}(\frac{U_i}{U})^{P}) \tag{5.3.16}$$

where U_i is the population of floor i and U is the total population above the main landing and j is the highest floor obtained starting at 1.

It can also be defined that the lowest reversal floor L can be estimated from any floor i with the probability that any of the passengers $P = K_{i\_\text{down}}(x)$ will not travel lower than the j^{th} floor is given by:

$$L = N_{i\_\text{down}} - \sum_{j=1}^{N_{i\_\text{down}}-1} \left(\frac{j}{N_{i\_\text{down}}}\right)^{P} \tag{5.3.17}$$

5.4 CAR CAPACITY PROBABILITY

The general TSP algorithm together with the VRP calculate the most optimal route for each elevator car to execute, based on all known passenger requests. These algorithms aim to generate a travel route which will reduce overall traveling costs and throughput maximisation. However as soon as the elevator car is dispatched, additional passenger requests are received as a result of continuous passenger arrivals. If the new requests are from a floor which is not on the same planned route as the current elevator car, the controller has to make a decision. Either the requests are added to the pre-planned route, or another elevator car is send. In most cases the new requests will be serviced by the nearest elevator car to reduce immediate transportation costs, but this will mean that the previously available elevator capacity has now been reduced and the original requests may not get serviced as planned. In this case the controller should make provision for the possibility of future requests in the planning stage, thus it should not allow the TSP algorithm to maximise the elevator car capacity for a pre-planned route.

An event C is defined as the elevator car reaching its full rated capacity during the execution of a pre-planned route when one or more additional requests are received during the trip. By taking into account the probability of reaching the car's capacity; it can result in a reduction in overall transportation cost and number of return trips or stops, but at the cost of a possible increase in the overall waiting time. The probability for event C, $P(C)$, to occur depends on its present available capacity and the calculated capacity at any point in the pre-planned route. The less available capacity there is the higher the probability, $P(C)$. It is noted that the available capacity is defined as the amount of additional passengers that the elevator car can transport, which translates to a theoretical number of additional requests that can be serviced. This translates to $P(C)$ which is calculated by summating the probability of receiving exactly zero requests plus the probability of receiving exactly one request, etc. up to the total number of requests that can be handled. Bernoulli trail probability, Equation: 5.2.1, can be easily utilised to obtain the required result. However the probability to receive a request must first be defined as well as the time period the probability is valid for.

In a 24 hour cycle, the controller should take into account the different traffic patterns and passenger arrival rates as accurately as possible.

5.4.1 UP-PEAK EXPECTED REQUESTS

In up-peak mode the various elevator cars in the group start at the main landing, delivering the passengers in the elevator cars to their destinations and then returning back to the main landing. All additional requests originating from other landing are ignored as far as possible to elevate the high passenger demand at the main landing. The expected requests during the up-peak mode will now be established based on exponential and independent passenger arrivals.

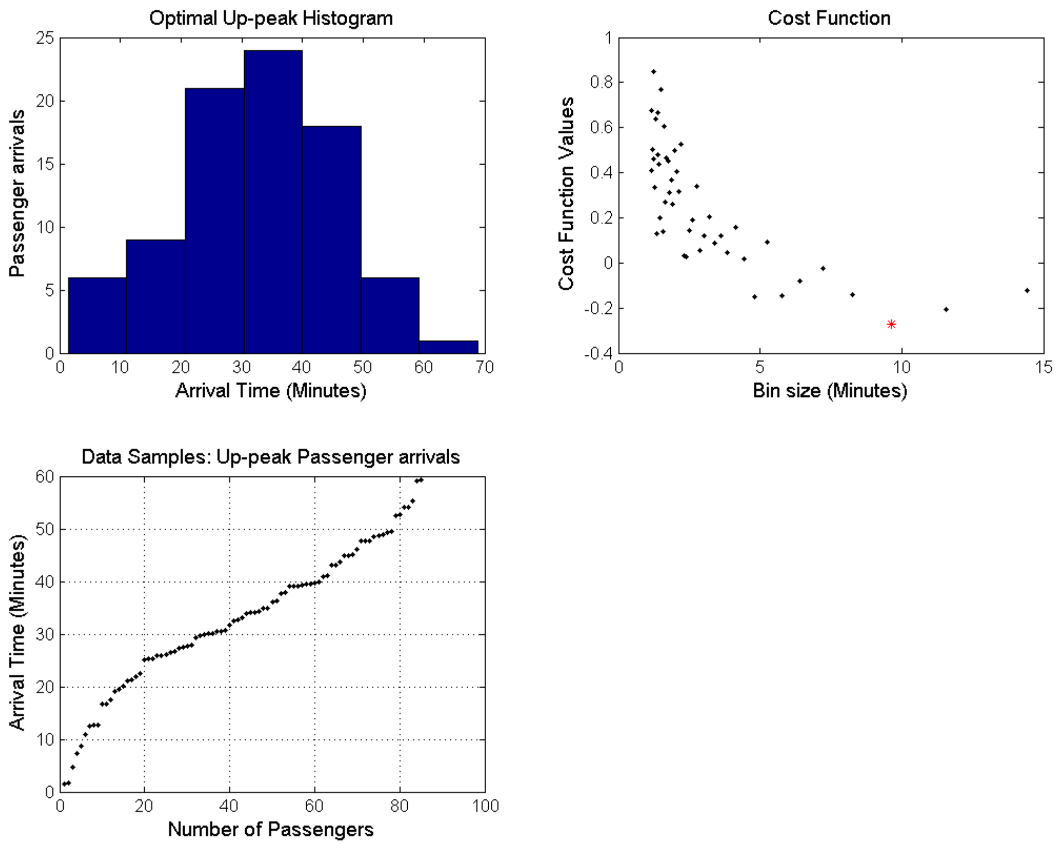


Figure 5.3: Actual non-homogeneous Poisson Passenger Arrivals (adaption from [10].

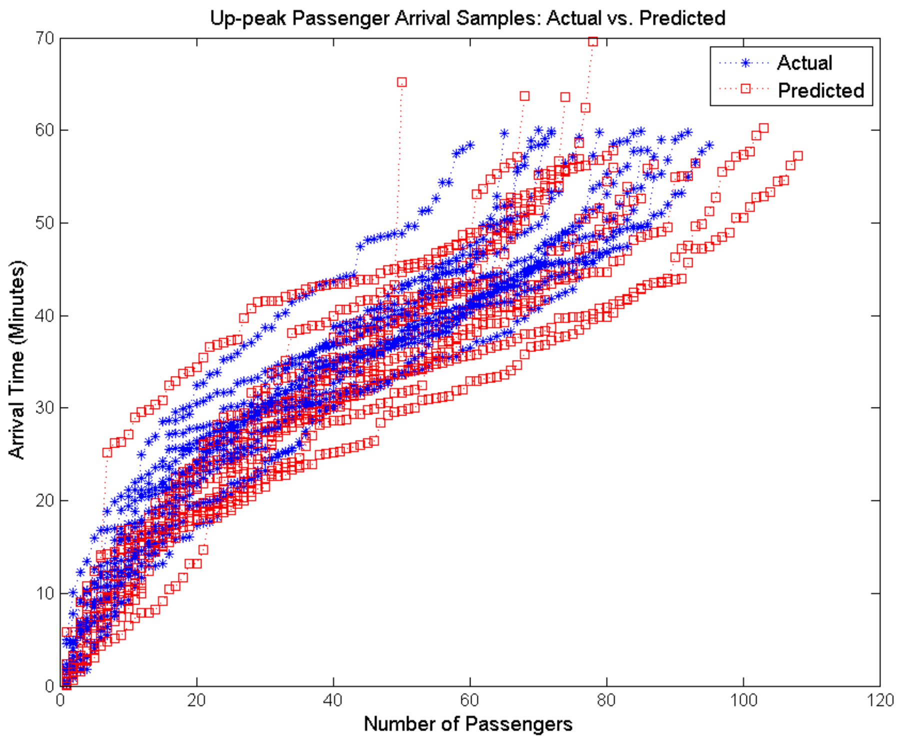


Figure 5.4: Up-peak Passenger Arrivals (Generated Actuals vs. Predicted)

During the up-peak mode the arrival rate varies as a function of time based on a traffic flow benchmark, taken from the Elevator Traffic Handbook written by Dr Gina Barney and is illustrated in Figure 8.1 [11]. However instead of defining the passenger demand in terms of total building population, our model will rather define continuous arrival rates based on the non-homogeneous Poisson process with time varying arrival rate $\lambda(t)$. The Poisson counting process, $K(t)$; $t > 0$ can be now be defined as a distribution of $\widetilde{K}(t,\tau)$ with the number of arrivals in an interval $(t,\tau]$ as follows [38]:

$$Pr\{\widetilde{K}(t,\tau) = n\} = \frac{[\widetilde{m}(t,\tau)]^n \mathrm{e}^{-\widetilde{m}(t,\tau)}}{n!} \tag{5.4.1}$$

Where

$$\widetilde{m}(t,\tau) = \int_t^\tau \lambda(u)\,du \tag{5.4.2}$$

The non-homogeneous Poisson process is thus defined with all the characteristics of a Poisson process over a non-linear time scale $K(t) = K(m(t))$ for each t [38]. The arrival rates will also change more abruptly in the peak traffic periods compared to the other traffic states, thus to get a better representation of the rate changes, the time period must also be shorted or increased together with the rate changes throughout the 24 hour cycle. The optimal time varying rate can be calculated with the use of spike density and spike rate estimation methods. In our case each spike refers to a constant Poisson rate for each predetermined time period. The predetermined time period are established by representing the spikes with the use of the Histogram method; the full range of the different rates are divided into N bins of width $\triangle$ and with the number of spikes s_i in the i^{th} bin. The mean s and the variance v are defined by Applegate [10] as:

$$s = \frac{1}{N}\sum_{i=1}^{N} s_i \tag{5.4.3}$$

$$v = \frac{1}{N}\sum_{i=1}^{N} s_i - s \tag{5.4.4}$$

With the cost function defined as:

$$C(\triangle) = \frac{2s - v}{\triangle^2} \tag{5.4.5}$$

By iteratively changing the bin sizes and the Poisson rates, the cost function should be minimised. The optimal bin size translates to the optimal time period that best suited the Poisson rates for each traffic period (traffic state). As illustrated in Figure 5.3 the passenger samples were generated with 12 different Poisson rates as a test case; which can be seen as the actual passenger arrivals. The non-homogeneous Poisson rates can now be optimally calculated by putting the actual arrivals through the spike rate estimator method. The result in this case is that we only required 7 different Poisson rates over the up-peak period of 60 minutes. The calculated Poisson rates can now be used to predict future up-peak passenger arrivals as illustrated by Figure 5.4. The expected non-homogeneous Poisson rates during the up-peak period can be improved by obtaining the average of all observed rates and continuously adjusting them. The rates can also be changed by means of a neural network after each up-peak period and by a sufficient amount of initial test samples. With 1000 iterations the calculated mean error between the generated actuals and the predicted number of passenger arrivals for the entire up-peak period was 8.64% with a variance of 43.16%. The same method can be followed for a scalable period of time, where the actual passenger arrivals are recorded and then represented by non-homogeneous Poisson rates with optimal histogram bin sizes.

A neural probability model will be developed later in Chapter 7.3, which will make use of the method presented in this Section. The optimal bin sizes of the non-homogeneous Poisson rates are used as input neurons to train the Neural Network to recognise and predict expected passenger requests for any traffic state.

5.5 SUMMARY AND CONCLUSIONS

In this chapter a probability model has been developed based on the probability philosophy definition and various theoretical predictions. The basic philosophy is to process previous collected traffic data and use it to estimate future traffic occurrences to create pre-planned elevator car responses to the probability of future passenger requests. The probabilistic traveling salesman problem (PTSP) was defined, which is similar to the classical TSP, but with service probabilities, assuming independence between floors. The PTSP is attempting to deal with the uncertainty of routing problems, like nondeterministic cost calculations and uncertainty in passenger demand at each floor.

References were made to well-known probability theorems like the Bernoulli trail probability and the Bayesian theorem that connects the respective prior floor probabilities with the posterior floor probabilities. The Poisson arrival probability function was implemented, with λ_i defined as the average number of passenger arrivals per floor in a specific period T. The general Poisson probability density $f(x)$ and the Poisson probability distribution $F(x)$ were used to provide a distribution model for stochastic passenger arrivals at each floor. Which means that the probability for a specific time is zero, however as a result of a repeating phenomenon called the Poisson process we are able to sufficiently describe the arrival patterns.

Through the non-homogeneous Poisson process the directional and up-peak expected requests could also be established. A technique was developed to obtain the best suited Poisson rates for each traffic period with the introduction of a cost function by Applegate [10]. The expected distance of travel and the expected car capacity probabilities also contributes to the TSP algorithm together with the VRP, to generate a travel route which will reduce overall traveling costs and maximise throughput. A neural probability model will be developed later in Chapter 7.3, in conjunction with the theoretical references from this chapter.

Chapter 6

WAITING TIMES CONSTRAINTS

6.1 WAITING TIME PHILOSOPHY

Waiting time is a good indication of the functionality of an elevator and can be used to compare different control algorithms and methods against each other. The basic philosophy of the controller is to find the most optimal route to be able to minimise overall waiting time of all passengers. Waiting times can be calculated from numerous compilations, such as from individual passengers or from a number of passengers that are grouped together that suit the same criteria. Groups of passengers that are easily classified together are everybody in the elevator car and at every landing floor at a moment in time. Other groups can be formulated from the passenger types: for instance visitors and employees. Also in regard to functional importance like managers, VIP guest, etc. Each passenger can also be represented by more than one group.

The philosophy of an intelligent elevator controller should not merely look at minimising waiting time at all cost, but rather at the concept of time management. By putting emphasis on time management; different priorities are taken into account and enforces a degree of compromise between them. For instance individual passengers or groups may have different degrees of acceptable waiting times as a result of an indifference or personal tolerance. The controller should base its decisions on these tolerances and adapt accordingly to user personalisation. Unfortunately the information required to distinguish between personal tolerances are not readily available and require an Expert System to establish unique passenger group waiting time requirements.

A few questions come to mind, in an attempt to create an Expert System with regard to waiting time tolerances for example: Are general workers more tolerant to waiting at an elevator than managers? Do guests show more patience than permanent personnel? Are passengers waiting at higher floors more accepting to longer waiting times than passengers closer to ground floor? Are people waiting at a floor that are close to their destination not the most intolerant to waiting times, because they have an alternative option of taking the stairs? Does the direction of travel have an influence on the acceptable waiting times? How does the number of passengers waiting at a floor have an influence on waiting time perceptions? Do waiting time tolerances differ for the same person or group at various periods, for instance at lunch time or day end?

6.2 BASIC WAITING TIME CALCULATIONS

At a basic level we want to minimise the waiting time of each passenger or group, subject to the space availability of the dispatched elevator. Thus we want to minimise

$$\sum_{j=1}^{N} c_j x_j \ , \tag{6.2.1}$$

subject to,

$$\sum_{j=1}^{N} a_{ij} x_j \leq b_i \ , \ (i = 1, 2, \ldots, M) \ , \tag{6.2.2}$$

with c_j, the cost (waiting time) for each allowed or not allowed (0 or 1) passenger x_j and a_{ij} the amount of available resources (up to M) for each passenger or group (up to N). Also b_i, the budget or the allowed time limit for each passenger or group. If more resources are required than $a_{ij} > 0$ or when the passenger or group free up some resources then $a_{ij} < 0$ in a period i. If the budget is increased then $b_i > 0$ and a reduction results in $b_i < 0$. The model can be further extended by allowing passengers from the same floor to be allowed to travel together, if $x_j \leq x_i$, and $x_i = 1$ then $x_j = 1$.

The total passenger waiting time (cost) can be estimated with the following assessing calculations per passenger or group N with a projected floor order list $\{0, \ldots, D_j\}$:

$$\Delta t_{\text{total}} = \sum_{j=0}^{D_j} (\Delta t_{\text{landing}} + \Delta t_{\text{car}}) \tag{6.2.3}$$

$$c_j = \sum_{j=0}^{D_{ij}} X_j((t_{\text{car arrive j}} - t_{\text{passenger arrive j}}) + (t_{\text{car arrive at } D_{ij}} - t_{\text{car depart from } D_{ij}})) \tag{6.2.4}$$

Equation 6.2.4 can be further defined in terms of the time spend in the car when the car is stationary ($\Delta t_{\text{stationary}}$) and the amount of time spend in the car while it is moving (Δt_{moving}) to each passenger's final destination (D_{ij}).

$$\Delta t_{\text{total waiting}} = \sum_{i=1}^{N} \sum_{j=0}^{D_j} X_{ij}(t_{\text{initial}} - t_{ij} + \Delta t_{\text{moving}} + (D+1)\Delta t_{\text{stationary}}) \quad (6.2.5)$$

Where t_{initial} is the initial time recorded for a pre-set floor order as estimated by the controller's dispatching method with j the order position in the order list. t_{ij} is the time recorded when each passenger arrive at the landing floor with X_{ij} the amount of passengers with the same arrival time $X_{ij} \leq 0$. j is the position in the order list when the passenger or group $(0, \ldots, N)$ will be picked up by the elevator car and D_j is the position in the order list where the destination floor is reached. Δt_{moving} is calculated by the total distance the car travels from the floor of arrival and the destination floor, times the rated speed of the elevator car.

6.3 WAITING TIME CONJOINING NETWORK

6.3.1 RADIAL BASE FUNCTIONS

The AI elevator controller is defined in such a way to make decisions across an environment where the decision making process is based on multiple objectives and constraints. The information with relation to the available alternatives to the satisfaction of the objectives are required and the weight of each objective must be defined. The end result from the various programmable and graphical methods or philosophies can then be compared against the baseline computational philosophy of numerous iterations based only on passenger arrive times with the same waiting time tolerances.

In an attempt to obtain a complete waiting time conjoining network or a representation of one, we will make use of radial basis function (RBF) techniques. An RBF network is based on the approximation of an arbitrary continuous function from accumulated data points or from linear superposition of localised basis functions [2]. The RBF network can be used for interpolation between waiting time data points from various individuals and groups and presented in a three dimensional space, neglecting time with $t = 0$. The purpose of a RBF network is twofold, the one is to obtain a function that provides exact interpolation between the input data points and the second to illustrate the influence of each data point on all the other data samples. Each individual passenger or any identified passenger group can be represented by different localised functions $\phi_q(x)$, with $q = 0, \ldots, N$ and N the number of functions.

The overall RBF network is constructed by the superpositioning of these functions, with weighing parameters, w_{kq} required to obtain exact interpolation for the test values $k = 1, \ldots, n$:

$$Y_k = \sum_{q=1}^{n} w_{kq}\phi_q(x) \tag{6.3.1}$$

Y_k represents a continuous differentiable surface, where every data point are passed through.

There are numerous base functions $\phi(r, r_0)$ to use, for example the multiquadric radial basis function:

$$\phi(r, r_0) = \sqrt{r^2 + {r_0}^2} \tag{6.3.2}$$

The inverse multiquadric radial basis function:

$$\phi(r, r_0) = \frac{1}{\sqrt{r^2 + {r_0}^2}} \tag{6.3.3}$$

The thin-plate spline radial basis function:

$$\phi(r, r_0) = r^2 \log \frac{r}{r_0} \tag{6.3.4}$$

The Gaussian radial base function:

$$\phi(r, r_0) = \mathrm{e}^{-\frac{r^2}{2r_0^2}} \tag{6.3.5}$$

Where r_0 is the scaling factor and r, the radial separation.

6.3.2 EXPONENTIAL RBF TEST MODEL

In an attempt to implement the philosophy, where the elevator cars in the elevator configuration are visiting the closest floors first, but also taking into account the total waiting times per floor. It can be illustrated by an example, together with radial base function techniques and a test function to influence the controller's decision making process. The Exponential RBF input values x and y, are provided in Table 6.1 with the exponential function output values z and summated interpolated values r. Each data sample $x, y, z = t(x, y)$ is represented with a base function $\phi_q(s)$ with $q = 0, \ldots, N$ and N the number of data samples. The Gaussian radial base function together with the exponential test function were used in this instance, because it proved to be the best suited for this application. The exponential test function $t(x, y)$ defined as,

$$t(x, y) = a_1 \mathrm{e}^{-a_2 x^{a_3} + a_4 y^{a_5}} + a_0\ , \tag{6.3.6}$$

Table 6.1: Exponential RBF Test Model Results.

Floor	x	y	$z = t(x,y)$	$r = r(x)$
1	0.1	0.001	0.990051	0.274
2	0.2	0.040	0.962343	0.307
3	0.3	0.080	0.919799	0.312
4	0.4	0.040	0.853508	0.302
5	0.5	0.0003	0.778801	0.297
6	0.6	0.0002	0.697676	0.309
7	0.7	0.080	0.61656	0.33
8	0.8	0.360	0.600255	0.34
9	0.9	0.400	0.522046	0.313
10	1.0	0.0001	0.367879	0.235

with $\forall x \in [0, \ \ldots \ , \ 1]$, $\forall y \in [0, \ldots, \ 1]$, scaling coefficients $a_1, \ldots, a_5 > 0$ and offset a_0, which were implemented over the input data, see Table 6.1. For this example the coefficients are initially set as follows: $a_0 = 0, \ a_1 = 1, \ a_2 = 1, \ a_3 = 2, \ a_4 = 1, \ a_5 = 2$, and then further changed iteratively to adjust the influence the input values have on the RBF system response. The y values represent the total waiting time at each floor, normalised across all floors and the x values represent the distance from the current position of the elevator car. The distance is given as the amount of floors from the elevator car divided by the total number of floors. For the purpose of this philosophy, the ratio between the distance to travel and the waiting time totals are minimised, in order to identify the most relevant floors to service which is also closer to the elevator car at any moment. The results is given by Table 6.1 and illustrated by Figures 6.1 and 6.2. The Gaussian radial base function was used to obtain exact interpolation between the provided data points with calculated weights: $w = [1.8233, -2.2001, 2.6798, -1.8873, 1.5579, -0.2822, 0.1047, 0.4144, 0.1244, 0.2816]$.

The variance or scaling factor σ, is set to 0.189, which is larger than the typical separation between input data points, but smaller than the maximum separation and is given by in Listing 6.1.

Listing 6.1: Gaussian Radial Base Function from Eq. 6.3.5

```
xyd = [ xd'; yd' ];
volume = prod ( max ( xyd , [] , 2 ) - min ( xyd , [] , 2 ) );
r0 = ( volume / nd ) ^ ( 1 / 2 );
```

Figure 6.1 illuminates the fact that the elevator car should service the closest floors and ignore the floors which are further, even if the highest waiting times are experienced at landings which are 8 and 9 floors away. In some cases the

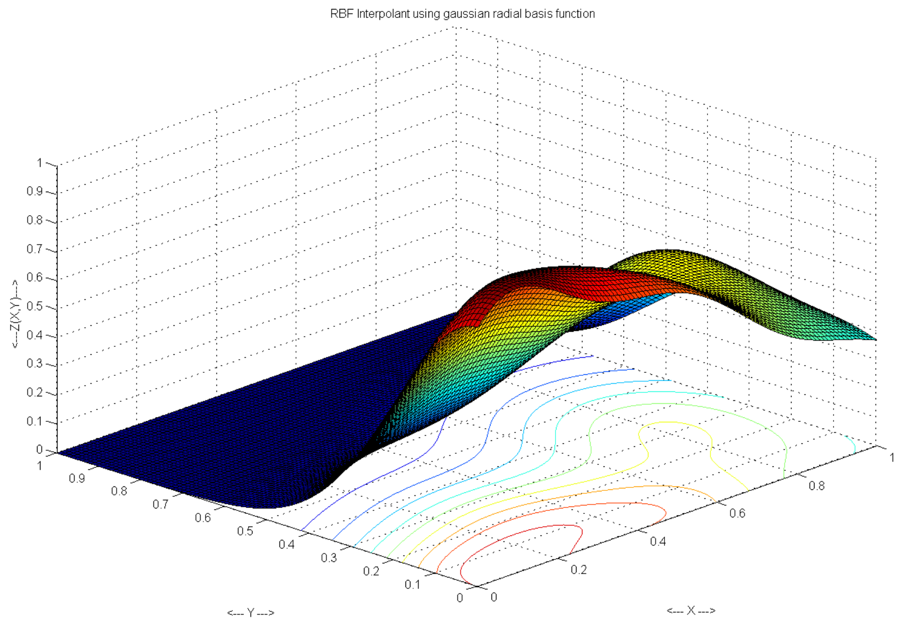


Figure 6.1: RBF Graphical Illustrations: Waiting Time vs. Distance from Elevator Car.

exact interpolation between data points fails to provide an intuitive reasoning method, but can be further improved by summating all the interpolated y values with the calculated radial base function. The result is a summated y value for every distance x, divided by the number of samples nix, taken from the RBF output values by $ZI()$ in Listing 6.2.

Listing 6.2: Interpolated Waiting Times.

```
nix = 100;
k = zeros ( 1, niy );
for col = 1 : nix
for row = 1 : niy
k(1,col) = k(1,col) + ZI(row,col);
end
end
k = (k./nix);
```

Figure 6.2 provides a few graphical representations from the exponential RBF test model results provided by Table 6.1. The $x -$ axis values represent the number of interpolated samples which can be divided by the total number of floors to obtain the distance from the elevator car. The $y - axis$ values represent the summated values divided by the number of interpolated samples

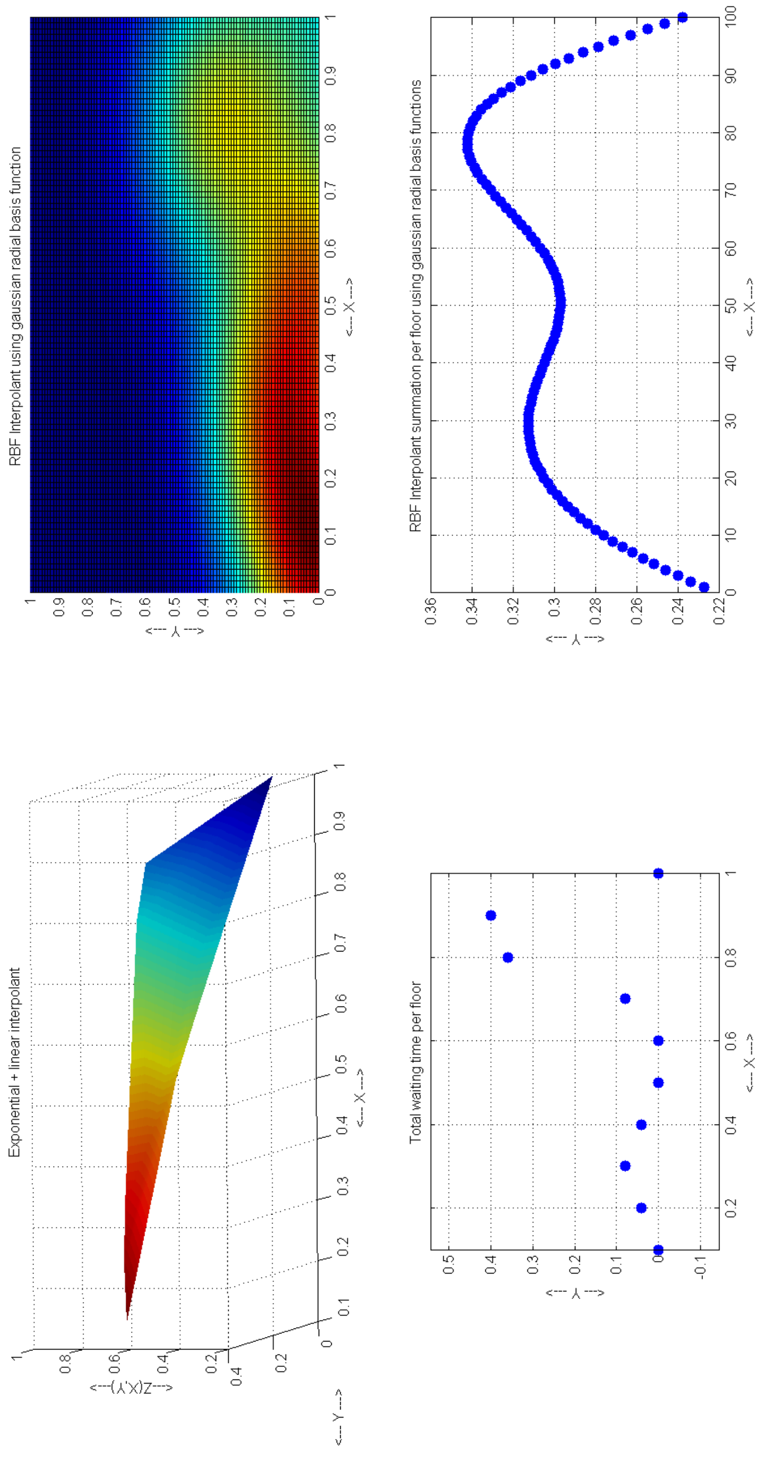


Figure 6.2: Exponential RBF Interpolant with Gaussian Radial Basis Function.

nix. In this case nix is set to a value of 100, which acts as a normaliser for the input values. The top left graph is plotted with reference to the exponential RBF with linear interpolation and the top right graph is the 2D Gaussian RBF representation of the results. The exact data sample points are plotted in the bottom left figure and the bottom right graph illustrates that the 8^{th} floor from the elevator car should get priority over any other floor, because of the accumulated influence from all the other data points in the near vicinity. In this instance the total waiting time at each floor had a bigger influence on the philosophy than the actual distance from the elevator car. By using curve fitting methods, like MATLAB's Basic Fitting Toolbox, the accumulated waiting times (y-values) can be obtained through a 6^{th} degree polynomial equation, $r(x)$ given by:

$$r(x) = p1x^6 \ + \ p2x^5 \ + \ p3x^4 \ + \ p4x^3 \ + \ p5x^2 \ + \ p6x \ + \ p7 \tag{6.3.7}$$

where $\bar{x}$ is centered at 50.5, $\sigma = 29.011$ and coefficients:
$p1 = 0.0073171$
$p2 = -0.004191$
$p3 = -0.054476$
$p4 = 0.0088988$
$p5 = 0.074638$
$p6 = 0.010553$
$p7 = 0.29758$.

Based on the results from Equation 6.3.7 and Table 6.1, the controller can priorities which floors to visit first and which to ignore. In this case the values are very close and indistinct, but other cases is clearer.

6.4 SUMMARY AND CONCLUSIONS

In this chapter the general waiting time philosophy was provided, which states that the controller should be able to minimise overall waiting time of all passengers by establishing the most optimal route to follow for each car. The concept of time management was also introduced, instead of just looking at time minimisation techniques. An equation was developed for any projected floor order list to provide the expected passenger waiting time and to be used in any TSP route costing calculations.

A waiting time conjoining network was developed, by implementing radial base function (RBF) techniques. An RBF network is based on the approximation of an arbitrary continuous function from accumulated data points or from linear superposition of localised basis functions [2]. The RBF network is used for exact interpolation between waiting times from various individuals and groups to illustrate the influence of accumulated waiting times across the building. The interpolated samples can be used by the controller to prioritise between which one of the floors to service next as a result of the accumulated influence from surrounding floors. It also resulted in the positioning of cars to be in close proximity to the next floor to service and the one after that. The RBF network can also be used for dynamic zoning when a clustering algorithm is implemented for destination dispatching purposes.

Chapter 7

NEURAL NETWORK ELEVATOR INTEGRATION

7.1 INTRODUCTION

When it comes to an elevator application, data is not always accurate or complete and it creates a case of imprecision. With numerous algorithms and methods available to dispatch elevator cars optimally, we do however still depend on the availability of passenger information to have a successful elevator control application. The issue of uncertainty is addressed with probability, fuzzy logic and Expert System theory as discussed in this Thesis, but also with the help of neural networks. Neural networks are most often implemented to deliver three types of solutions, which is classification, pattern recognition and prediction.

The elevator controller application has a few classification requirements throughout data processing; for instance the present state of the elevator and the available car capacity can be classified against pre-set arrangements, see Section 7.4. Other elements that requires classification will be the respective traffic patterns and passenger demand intensities during a time period, see Section 7.3. By using membership functions we can classify the fuzziness or imprecise approximations of any discreet or continuous elements. Membership functions can be obtained through numerous ways, for example by intuition, inference, rank ordering, inductive reasoning, etc. [39]. Neural networks and genetic algorithms are also effective ways of obtaining membership functions. Well trained neural networks can be used for membership mapping of any of the data points and can be effectively checked against a test set. GA is used to obtain membership functions through obtaining the solutions with the best fitness levels.

Neural networks are usually required for applications that cannot be solved by general logic programming functions. However in order to investigate a more adaptive model of control, it requires the aid of machine learning techniques. With changing traffic patterns it is important that the system continuously produce and implement better solutions to deal with the changing environment. Neural networks are implemented in Section 7.2 as a pattern recognition solution to develop an accurate energy model representation together with accurate running time modeling against different distances and load variations. Neural Networks are also utilised to model the passenger distribution across the building and to predict passenger demand throughout any scalable period of time. The prediction model is created in Section 7.3. This chapter also introduces automated reasoning principles in Section 7.5. The possibility of an online testing platform can also be investigated, which can learn any new algorithms and automatically reconsolidate when or if such algorithm should be implemented to improve the ultimate performance of the controller.

7.2 NEURAL NETWORK ENERGY MODEL

Every elevator is unique in terms of its configuration, energy signatures and location. Theoretical variables are not the same from one elevator to the next, with a lot of uncertainties like tension, friction, energy inefficiencies with mechanical, thermal and energy losses. This means that a theoretical model is very difficult to obtain and trending against various load conditions is not always obtainable. However neural network techniques are effectively used in this section to establish an accurate energy model representation of the elevator application. The model is still unit specific, but is easier to obtain and provide the best possible results.

7.2.1 ENERGY SAMPLE EXTRACTION

When creating an energy model; it is paramount to analyse the machine together with the changing load conditions. For the implementation of a supervised neural network learning algorithm; each sample representation that relates to a specific load condition is recorded and stored. Energy sample extraction can be done by a trigger function, where the data recorder is activated with a predefined activation event. In this case, the data is recorded at the start of a current spike of 50% the rated current and stopped at 10% for a duration of 1 second. The current signatures together with the accompanied voltage signatures provide sufficient information about the system response to the recorded load conditions. With association with the current and voltage samples, we can further develop accurate time, power and energy consumption models to be presented to either new and independent neural networks or to have additional neurons added to the existing neural network output layer structure.

Each NN model can represent each sample as a whole, from the trigger point to the end of the sample, or it can be divided as illustrated in Figure 4.14. The benefit to dividing each sample into different sections are to provide more information about the system response. It is possible that certain sections will not have the same rate of change as other sections for different load conditions or previous hidden patterns can now be successfully extracted. To divide each sample into sections are normally completed by additional triggers to identify each notable amplitude change or to set a duration set point for each section. However each of these approaches will proof to be unsuccessful with a dynamic system where the system response is unknown during the sample extraction phase. When the Expert Knowledge about the system is not available we can implement a classification technique used in various machine learning applications, called k-means classification. It allows us to choose the number of sections or clusters, defined by the k variable, where each recorded data point in the energy sample are sorted or collected in the various k clusters they belong to. The benefit to this technique is that it can be implemented in any elevator application and it is not only subjected to a specific elevator configuration. It is important to have an adaptable system that can extract samples from any elevator configuration it is subjected to, in order to have a degree of artificial intelligence and to allow it to create its own neural network as accurately as possible.

The original samples were taken at a sampling frequency of 100 000 hertz which were effectively divided between 6 data sampling channels (3 current and 3 voltage probes), thus we have access to 16 666 data points per second per channel. The acceptable sampling frequency f_s should be greater than double the signal bandwidth B, to avoid any sampling overlapping repetitions [40]. The minimum sampling frequency is known as the Nyquist rate with the corresponding sampling interval; $T_s = \frac{1}{2B}$ called the Nyquist interval [40]. Thus we can down-sample the taken measurements by a factor of 160, which is still above the Nyquist rate of 100 hertz in this specific application.

The k-means energy sample extraction process is illustrated with Figure 7.1. During the initialise phase, the coordinates of the clusters are uniformly distributed across the total duration or total number of data points. The `kCluster_allocation()` method then divides the provided data points between all the clusters, based on the cluster which is the closest distance from the data point amplitude and keeps true to the data input pattern. After every data point is allocated to a cluster, the coordinates of the clusters are re-defined to minimise the total distance from all allocated data points. After all data points have been allocated, the `kMeans_cluster()` method shuffles

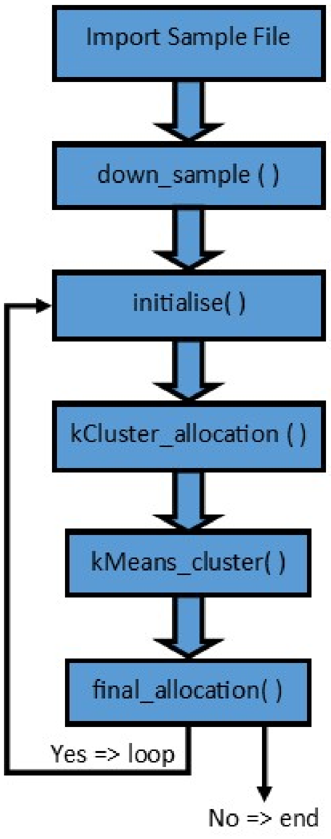


Figure 7.1: K-means Energy Sample Extraction Process.

the cluster positions as required, while calculating the means of every group of points after each change. The method iterates until equilibrium occurs between all clusters or until the mean values have settled.

With `final_allocation()` method the final cluster coordinates are fed back to the `initialise()` method and used as "data points" for the process to repeat. All allocated data points from the previous cycle are now represented by the number of cluster positions used as new input data points. Initially a large number of clusters are chosen and then by each iteration reduced, until the required accuracy is obtained between different sections of the energy sample. In this case all the data points with similar rms values are grouped together, with the constraint that data points are not to be taken out of position, to retain the input pattern. From the voltage sample channels it is noticed that only a single cluster is left after process execution, because the entire energy sample can be defined with only one rms value across all data points; normally in the range of 220 to 230V. The current channels were accurately divided between starting current, acceleration or deceleration and rated current stages.

7.2.2 DATA PREPARATION

In order to create an artificial neural system based on the canonical neural computations of the human brain, it is paramount to replicate how the brain process sensory input information. Based on studies from Carondini and Heeger [41], the brain uses exponentiation and linear filtering as a form of pre-processing information. They stated that: "exponentiation are a form of thresholding which allow the brain to maintain sensory selectivity, decorrelating signals and establishing perceptual choice together with linear filtering that refers to the weighted summation by linear receptive fields" [41]. Provision is made for these observations by implementing 3 types of threshold or activation functions, namely the Linear, Sigmoid and the Tanh functions. These activation functions are used to scale the output of each neural layer into the desired domain. The Sigmoid function only allows for positive values to pass through and the Tanh function allows for both negative and positive values to the output. The last two named functions allows the neural network its non-linear capabilities in scaling the different weight functions as required to minimise output error. When the structure of the neural network is decided upon, including the number of layers, activation function and type of neural network, the input data together with the output training data needs to be prepared to fit into the same domain. In this instance the input and output data extractions are normalised to be in the range between -1 to 1 or between 0 to 1, depending on the threshold function.

As with exponentiation and linear filtering of the human brain, there are a few normalising methods to implement, which are in reality only scaling mechanisms to be applied to the training data. These scaling techniques include: Centralised scaling [42]:

$$X_{i,-1\ to\ 1} = \frac{X_i - \frac{X_{\max}+X_{\min}}{2}}{\frac{X_{\max}+X_{\min}}{2}} , \tag{7.2.1}$$

where X_i are the i^{th} data point, X_{min} and X_{max} the minima and maxima among all data points respectively and $X_{i,-1\ to\ 1}$ the data points normalised between -1 and 1. MaxMin scaling can be completed with:

$$X_{i,0\ to\ 1} = \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{7.2.2}$$

Maximum scaling refers to each data point divided by the maxima among all data points, which are further multiplied by 0.9 to represent the 90% Maximum scaling method or scaled together with an offset. It is often useful to represent data in the 0.1 to 0.9 range, to avoid the neural network of reaching saturated weights at the minimum and maximum input values. An Expert System scaling can also be implemented; where some pre-knowledge about the model is available and is used to create a bias in the input data set as required. In our application the load input variable is biased around 0.5 for a balanced load, where the counterweight is perfectly balanced with the car and its load. As the elevator configuration becomes unbalanced in the direction of travel; the variable reaches a maximum of 1 or if unbalanced against the movement of the car; the input variable are reduced to a minimum of 0.

Carondini and Heeger [41], also referred to a third type of canonical neural computation, namely divisive normalisation; where the human brain computes a ratio between the response of a single neuron and the summed activity of a pool of neurons. To replicate this phenomena, 2 additional normalisers are introduced, namely the Z-value method and the Multiplicative normalisation. The former is defined with [42]:

$$X_{i,\sigma} = \frac{X_i - \overline{X_s}}{\sigma_{x,s}} \tag{7.2.3}$$

where X_i are the i^{th} data point, $\overline{X_s}$ and $\sigma_{x,s}$ the average and the standard deviation of all the sample data points respectively. $X_{i,\sigma}$ provides a Z-score; which reflects how many standard deviations from the average each data point falls. Multiplicative normalisation is where each input X_i is scaled by a normalisation factor [3]:

$$f = \frac{1}{\sqrt{\sum_{i=0}^{n-1} X_i^2}} \tag{7.2.4}$$

Table 7.1: Output Normaliser Error Performance: Neural Network Output vs. Training Data.

Output Normaliser		I_{fullD}	T_{fullD}	P_{fullD}	E_{fullD}
Maximum	Value:	1.06 A	1.01 Sec	812.9 W	17040 J
	Percentage:	2.67 %	3.82 %	3.66 %	4.16 %
90 % Maximum	Value:	0.84 A	0.85 Sec	554.41 W	13758 J
	Percentage:	2.12 %	**3.24 %**	2.5 %	3.36 %
MinMax	Value:	0.79 A	0.91 Sec	433.74 W	11981 J
	Percentage:	**1.99 %**	3.43 %	**1.96 %**	**2.93 %**
Z-value	Value:	1.27 A	1.9 Sec	726.7 W	23975 J
	Percentage:	3.2 %	7.21 %	3.276 %	5.86 %
Multiplicative	Value:	1.290 A	0.95 Sec	550.34 W	15114.5 J
	Percentage:	3.27 %	3.61 %	2.48 %	3.69 %

The various normalisation functions referred to in this section can be directly applied to the training data, however some normalisers perform better than others as can be expected. With Table 7.1 the different output normalisers are compared with the use of a common input normaliser, namely the Expert System normaliser. The root mean square (RMS) error, defined by Equation 7.2.5, is calculated for the data training set to obtain the rate of error based on the ideal results [3]:

$$X_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} X_i^2} = \sqrt{\frac{X_1^2 + X_2^2 + X_3^2 + \cdots + X_n^2}{n}} \tag{7.2.5}$$

The RMS error percentage of the current I_{fullD}, power P_{fullD} and energy E_{fullD} output neurons are minimised with scaling normalisers. The input variables are defined for the full duration (fullD) of a sample and are not divided into clusters in this case. Before normalising, the values are close to each other, which reduces the rate of change between the input values. By implementing the MinMax normaliser; the output value ranges have been widened to maximise the sensitivity to the input changes. The duration output neuron T_{fullD}, also performs better by scaling the output values rather than standardising them. The duration output neuron have a near linear response and should only be resized to fit into the required neural network domain. The z-value normaliser should be used when the mean and the variance of a data set is more important to the system response and often stays constant for any stimulus. The multiplicative normaliser is optimal if the data points are all close to zero, where other normalisers will allow the synthetic component of the input to dominate the smaller values [3].

The input normalisers are summarised by Table: 7.2, where the minimum, maximum, average and standard deviations are provided after all input training data have been normalised by each normaliser. The benefits to normalising the input variables are to allow all input variables to have the same dimensionality and to be able to reduce input redundancy. It is important when a neural network have multiple input variables, that one input variable does not have an overbearing effect as a result of its larger values over variables with smaller values. With reference to LeCun et al. [43], it is optimal to have the covariance of the input variables the same and to have the variables uncorrelated if possible. With uncorrelated variables, the learning rate is faster, where each variable have an independent effect on the output and this means that the input neuron weights can also be changed independently. LeCun et al. [43], also make mention that if the input variables are close to zero then the convergence rate is usually faster.

The Centralised normaliser performs on par with the Expert System normaliser, because it makes good use of the extended range between -1 to 1. Because a positive input coefficient can allow for a positive response to the output variables and vice versa. In the elevator application the Centralised

Table 7.2: Input Normalisation: Training Data Pre-processing.

Input Normaliser		Direction (I1)	Distance (I2)	Load (I3)
Expert System:	Max	1	1	0.71
	Min	0	0.2	0
	Average	0.5	0.5	0.3
	StDev	0.5	0.29	0.19
Centralised:	Max	1	1	1
	Min	-1	-1	-1
	Average	0	-0.25	-0.17
	StDev	1.01	0.72	0.53
MaxMin:	Max	1	1	1
	Min	0	0	0
	Average	0.5	0.37	0.41
	StDev	0.5	0.36	0.27
Maximum:	Max	1	1	1
	Min	0	0.2	0
	Average	0.5	0.5	0.41
	StDev	0.5	0.29	0.27
Z-value:	Max	1	1.75	2.2
	Min	-1	-1.04	-1.55
	Average	0	1.4E-16	-2.15E-16
	StDev	1	1	1
Multiplicative:	Max	0.15	0.19	0.22
	Min	0	0.04	0
	Average	0.08	0.09	0.09
	StDev.S	0.08	0.05	0.06

normaliser accurately apply the different coefficients to the weight imbalances and the direction of travel. Thus an input Expert System normaliser is not crucial to have in this kind of application.

7.2.3 SAMPLE REPRESENTATION

The total amount of combinations of the input variables can be seen as a vector space that form a n dimensional hypercube, where features are vertices and form connections within the interconnected network. To train the network, a percentage of data points is required to represent a large portion of the hypercube. The edge length $e_n(p)$, from a fraction of data points p, within a n dimensional space can give an indication of the effected space, and is defined with [44]:

$$e_n(p) = p^{1/n} \tag{7.2.6}$$

When the total amount of vector combinations, m are known, the distance between data points can be calculated with [44]:

$$d_n(m) = \frac{1}{2}(\frac{1}{m})^{1/n} \tag{7.2.7}$$

It is ideal to have the data points closer to each other rather than to the edge of the sampled data set, where connections are not well formed. However the amount of training samples should be a good representation of all possible combinations, but should be limited to avoid the neural network from building the required connections and predictions for missing data points. The neural network is also seen to train faster with limited training data points, at the cost of accuracy. Also as a result of the dominant waveform created by the neural network it is easier to classify outliers from the training data. Outliers are data points that don't fit the general predicted pattern and can be a result of measurement mistakes or from out of the ordinary conditions. Outliers should be removed from the training set, if the neural network seems unstable and not capable of filtering these anomalies automatically.

7.2.4 NEURAL NETWORK ENERGY MODEL TRENDING AND CONCLUSIONS

The sample data that was used to plot Figure 4.15 in Section 4.4.1 was obtained by separate measurement probes to obtain the 4 pole from the 18 pole winding supply current. However the clustering method described in Section 7.2.1 successfully extracted to 2 datasets from the main supply current samples as well. Thus the k-means sample extraction method can be utilised to extract certain sections from a main supply point as an alternative to any additional measuring points normally required. The developed k-means sample extraction method can also be used with the latest regenerative elevator configurations, to establish accurate energy saving declarations from its regenerative capabilities. This method can easily distinguish and extract a separate dataset for every motor state provided by Figure 4.11.

With reference to Figure 4.16 the elevator supply current (in A_{rms}) and power consumption (in kW) were plotted against each respective travelled distance and load percentage of the rated car capacity. The red and orange data points represent the actual measured data and the blue and green continuous lines are from the output neural layer. It is clear that the predicted values resemble the actual measurements closely and that the neural network trending capabilities are successfully utilised to model the application accurately with different load conditions. The same can be concluded with respect to the predicted running times provided by Figure 4.17.

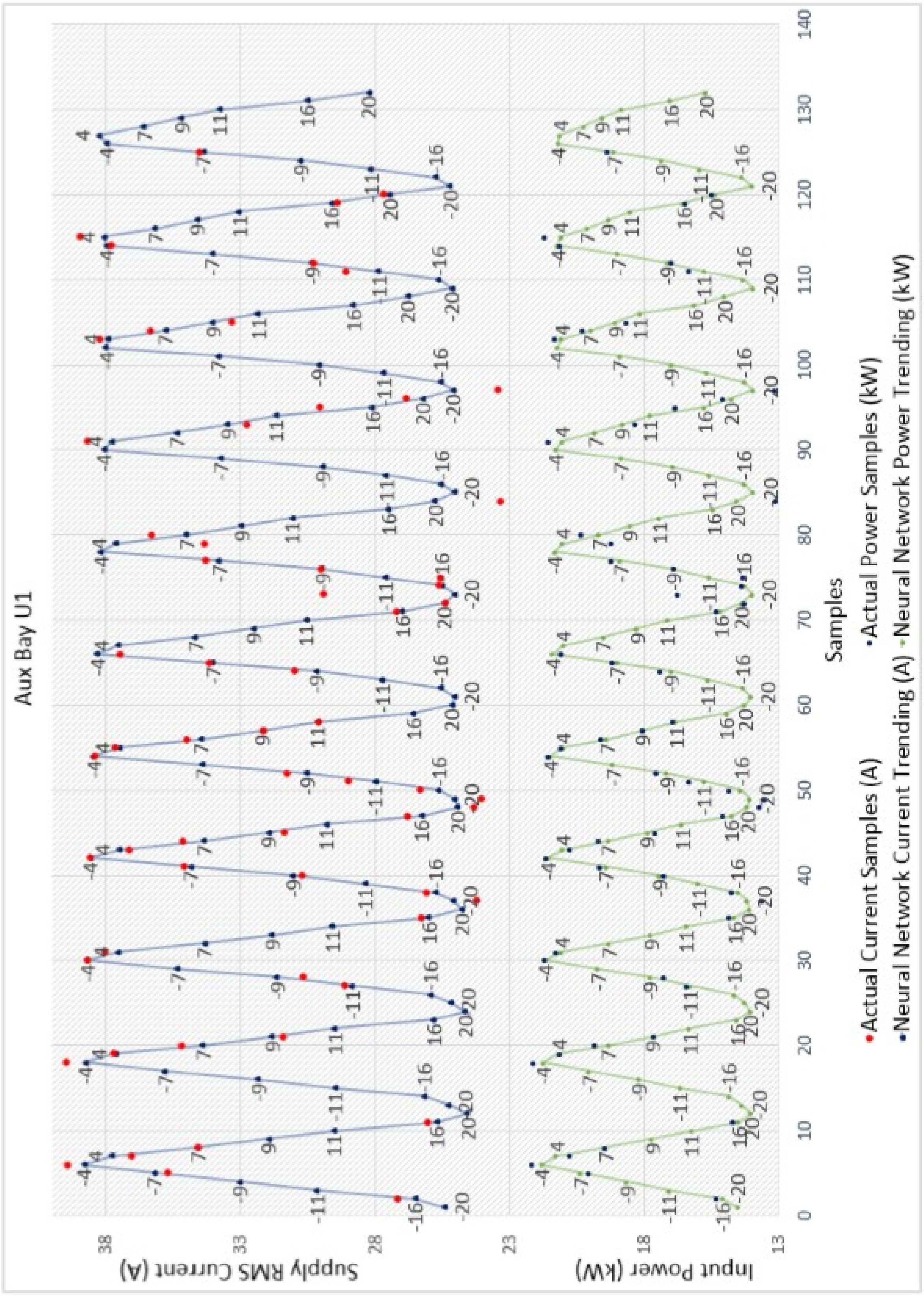


Figure 7.2: Neural Network Supply Current and Power Consumption Pattern Recognition.

The three input variables which were used to train the elevator energy signature neural network are direction, distance and load. Each variable has a distinct effect on the elevator system response prediction. The first variable is of the categorical type, presented as a numeric binary set with 2 values. The second and third input variables are of the numeric type and has an order and a distance relation [44]. From the different energy samples it became apparent that we are not dealing with time-dependent data, but rather distance dependent. The maximum and minimum distances being traveled between floors, create a local maxima and minima in the predicted waveforms. The elevator's rope sheave creates a uniform circular motion projected in the vertical plane as a simple harmonic motion. This occurrence is similar to the dynamics of a pendulum or an extended/retracted spring mechanisms; where a sinus waveform is created when plotted against distance travelled. Thus it can be stated that the distance input variable dictates the general pattern of the predicted data, as seen in Figure 7.2. From the previous mentioned figures it can be seen that the direction and load input variables have an effect on the trending of the output data, rather than the waveform itself. The trending that occurs between the maxima and minima is justified by the physical relationships of the elevator configuration as discussed in Section 4.4.1. Each relative distance travelled can be seen changing from one cycle to the next as the load increases from left to right in Figure: 7.2.

7.3 NEURAL NETWORK PASSENGER DEMAND AND TRAFFIC STATE CLASSIFICATION

7.3.1 INTRODUCTION

The concept of probability was introduced theoretically with reference to Chapter 5, where the Poisson arrival distribution was presented. We also made reference to the Probabilistic Traveling Salesman Problem (PTSP) with directional and expected travel distance probabilities. We will now develop a fully functional probability model, to be integrated with the Intelligent Elevator Controller. Neural Networks are utilised in this case to model the passenger distribution across the building and to predict passenger demand throughout any scalable period of time.

7.3.2 NN PASSENGER ARRIVAL MODEL

With computer based controllers it is not difficult to store all incoming requests in a centralised database, together with the timestamp, arrival floor, destination and passenger identification if available. Past requests can then be recalled from the database for a period of time and reconciled into a passenger demand distribution. The passenger arrival data points are summarised into representative non-homogeneous Poisson rate intervals, with corresponding cost function, mean and variance calculations by Equations 5.4.3, 5.4.4 and 5.4.5. When the passenger arrivals are extracted from the database, they can be grouped according to similar characteristics for instance: per floor (arrival and destination floors separately), direction, per zone (floors grouped together) or personal divisions. The result is optimal histogram representations of the passenger arrivals per group.

A Neural Network is developed for each histogram representation with the following procedure: A number of input neurons is defined, let's make it 5 input variables for example. The input neurons will each receive a value of a single histogram interval value, taken in the corresponding order starting from the beginning of the extracted period. The output neuron receives the first value after all the input neurons have been given values. In this case it is the 6_{th} interval value. Thus the NN has just learned what the value is after the specific sequence of values are provided. This procedure is iterated throughout the histogram length by shifting the input and output neurons with an interval each time. When all histogram intervals have been entered into the NN, this correlates to one training epoch and should be repeated for a number of epochs or if the required error rate is less than 5%.

As a result of the learning procedure the required traffic patterns have been formulated and can be used to predict future passenger arrival rates accurately. In Figure 7.3 it can be seen how the elevator prediction model has learned a 5 day period and became very accurate in predicting future passenger arrival rates. This specific NN was trained with only the directional characteristics of the passenger arrivals summated across all the floors. Various other predictive networks have also been created to increase overall building traffic demand expectations and are summarised as follows:

- Per floor traffic distributions.
- Zone or cluster distributions.
- Calendar predictions.
- Personnel and visitor distributions.

The next step is to incorporate the predicted passenger demand distributions into the decision making process of the Intelligent Elevator Controller.

7.3.3 TRAFFIC STATE PREDICTION AND CLASSIFICATION

With the assistance of an Expert System it is possible to accurately predict the building 's passenger demand distributions and design the control philosophies accordingly. However for most elevator installations a typical controller is installed, with pre-programmed generic control philosophies. The generic control settings are inherently linked to the different elevator states and have pre-programmed respond cards that link an algorithm to a traffic state. Figure 8.1 in Section 8.1.2 is an example of such an Expert System that triggers the controller's decision responds. Thus it is important to establish the traffic state for the elevator group as fast as possible in order to trigger the pre-programmed control algorithm. Without a prediction model the typical controller actually recognises an increase in passenger demand and then reacts accordingly, but often the traffic state has changed a while ago. With the use of our NN prediction model we were able to predict when the traffic state is just about to change or at the exact moment it alters. Thus the controller have a faster response time to traffic pattern changes. With our on-line model it is possible to predict the so-called up-peak state within one Poisson rate interval, which in our case was set to 3 minutes; where all elevator cars were immediately directed to the landing floor.

A second advantage of a predictive model is to make confident alterations to the generic control algorithms, in order to optimise elevator group control. The different networks are used coherently to define a new approach to conclude dispatching decisions based on more relevant past experiences. The new approach is able to change the control philosophy from week to week and respond faster and more effectively to changes in the traffic patterns throughout the day. The competitiveness of the adaptable control algorithms improves over time as it comes closer to optimal offline computational routes. The adaptable control concept is explained as follows by means of the destination dispatching (DD) algorithm that is triggered by the up peak state. The DD algorithm as illustrated in Figure 8.11 did extremely well compared to other control philosophies. The generic DD algorithm divides the building levels into a number of equal zones from the landing floor, which correlates to the number of cars in the configuration. Each car is then restricted to only travel between its allocated zone and the main landing floor. In the plotted figure, the simulated building population was equally distributed across all floors, which best suited the generic algorithm. But if a real life situation is created where the building's population is only occupying 60 % of the building capacity, with the

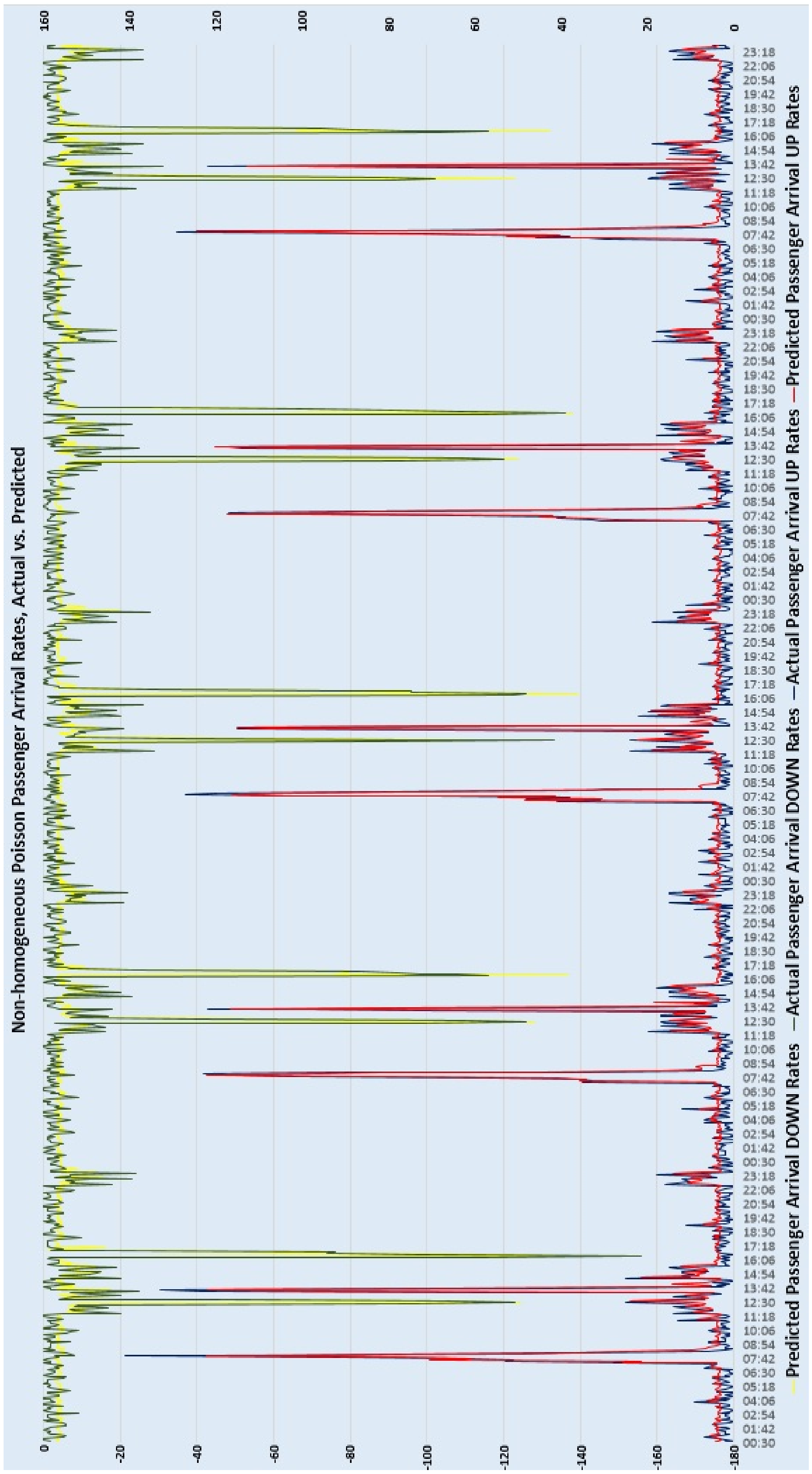


Figure 7.3: Non-homogeneous Poisson Passenger Arrival Rates, Actual vs. Predicted for a 5 Day Period

rest of the floors used as storage, the performance of the generic DD doesn't look that promising. By implementing the generic DD in this case; resulted in multiple cars not being utilised, because the allocated zones was never serviced.

A situation can also be described for industrial type buildings, where you have personnel shift change periods, which lasts up to an hour. The typical Expert System doesn't make provision for these additional high traffic demand periods or any other out of the norm traffic patterns. This means that there are no generic response to optimise the elevator group control. However any passenger demand changes can be quickly learned and adequately predicted by a developed Neural Network prediction model without any user input. The developed NN model had a direct influence on the intelligent elevator controller's performance improvements. This proves that any out of normal passenger demand can be directed into a pattern and used to alter the control algorithms accordingly. If a pattern cannot be established, it means that the out of normal passenger demand should be treated as an anomaly and not influence future control decisions.

7.4 CAR CAPACITY CLASSIFICATION

7.4.1 SMARTSTART EXPERT SYSTEM

The 3 online-dial-a-ride strategies that were presented in Section: 2.2, were effectively implemented in our elevator control application. The Smartstart strategy allows the server to respond to new requests and replan the optimal route, based on all known information. Also it has the ability to deny any new requests and effectively execute an ignore demand. The optimal point between replan and ignore demands were obtained by trial and error for various elevator configurations and for different amount of cars in an elevator group. The different conversion points from Replan to Ignore are plotted in Figure 7.4, and it illustrates the effect on the total accumulated waiting time as a percentage of the maxima. The results seems relevant to most configurations that were tested. The optimal range for minimum overall waiting time is when the car load is between 20 % and 40 % of the rated full capacity. This suggest that the Smartstart strategy setting should always be in this range, however with additional system requirements; it becomes more complicated than to have a default set point.

For optimal passenger comfortability in the elevator cars, the capacity set point should be kept to a minimum. A low set point will also reduce the number of stops per passenger from the origin floor to their respective destination floors. In dense traffic populations it is even possible to allow no additional stops before arriving at the destination floor, and can be easily implemented

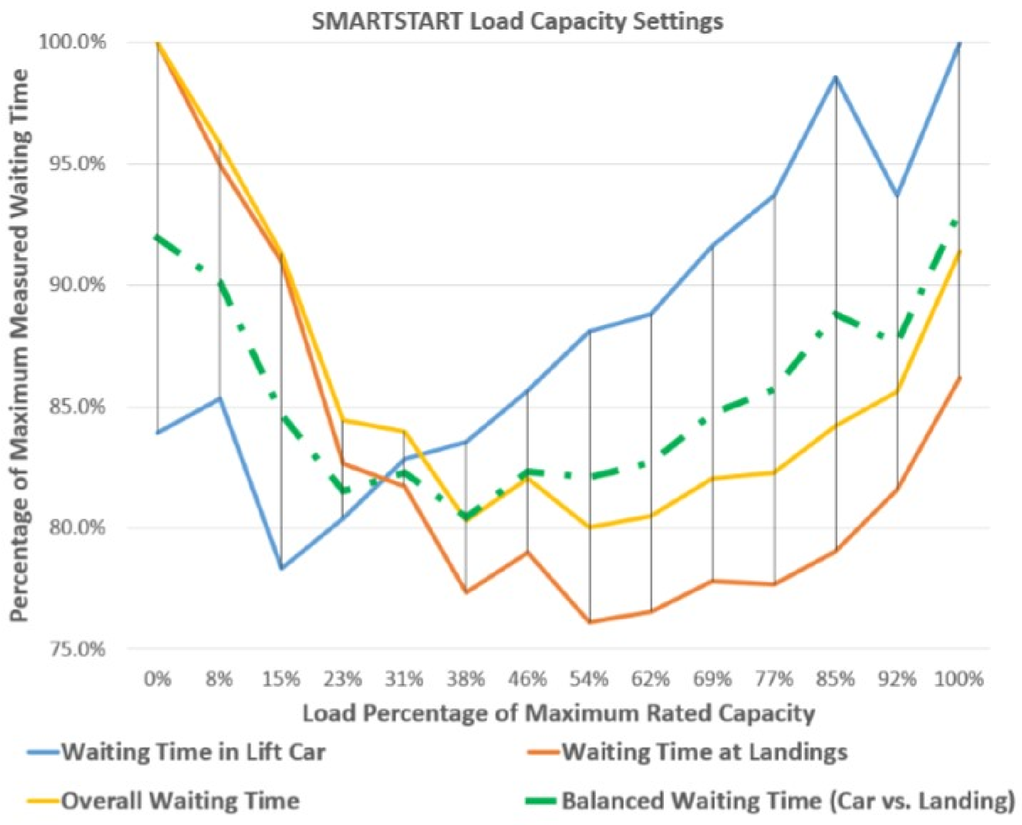


Figure 7.4: Smartstart Load Capacity Setting's Effect on Overall Waiting Time

by the mentioned Smart-Start setting. During high traffic periods, like up-peak and down-peak traffic patterns, the landing waiting time is reduced by adjusting the capacity set point to a range between 50 % and 70 % of the rated capacity. The accumulated time spend in the elevator cars will increase, but more passengers are serviced at a time at the cost of personal comfortability.

7.4.2 NEURAL SMARTSTART MODEL

A general Smartstart Expert System were developed in Section 7.4.1 for typical elevator configurations and for most control algorithms. However an online Neural Network enabled Smartstart system can be developed to know the most optimal Smartstart settings for the given traffic pattern and control algorithm. A specific algorithm namely P1-C, was tested for a 24 hour cycle, 5 cars, 15 floors and a population of a 1000 people, as illustrated by Figure 7.5. The figure illustrates that the optimal setting is not the same throughout the 24 hour cycle and should be changed dynamically to improve overall performance. The optimal conversion points from Replan to Ignore was in contrast to the Expert System from Figure 7.4 in terms of Ignoring any new requests if there are 2 or more people in the car. After the up-peak traffic state the optimal

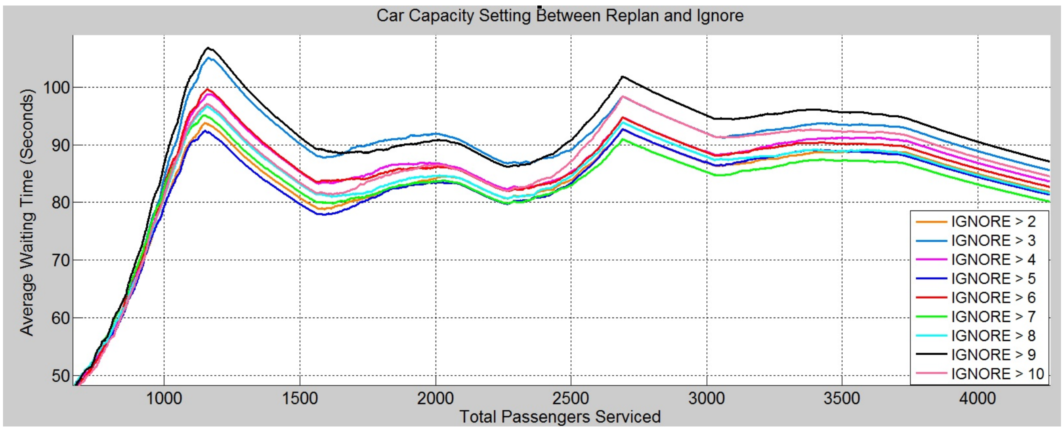


Figure 7.5: Dynamic Smartstart Load Capacity Setting's Effect on Waiting Time

conversion point is 7 and correlates to the Expert System however, 6 did not perform well but was in fact a recommended value.

The Smartstart Neural Network is trained offline from the actual service request database or via the simulated traffic generator. The optimal car capacity set point is then calculated iteratively for a scalable duration by adjusting the settings and comparing the output performance. To improve accuracy, the car capacity set point is calculated for every point in time due to changing passenger demands and traffic states. This results in a NN that is trained to deliver the best Smartstart setting for every Poisson passenger arrival rate. The Intelligent Elevator Controller should train a Smartstart NN for every new control algorithms that it receives and update it regularly when more recent passenger demand patterns becomes available. When every control algorithm has its own Smartstart NN stored on the main drive the system is ready for online operation.

7.5 AUTOMATED REASONING WITH NEURAL NETWORKS

7.5.1 INTRODUCTION

The process of making decisions based on past experiences translates to the well-known field of case base reasoning (CBR) in AI. CBR is mostly based on regulation and repetition principle, which states that certain actions will be taken again under the similar conditions and would produce similar results [27]. CBR in turn also define Expert Systems, usually based on human expert knowledge, but we would rather look at past control decisions instead of implicitly defining every condition. Thus the unique learning abilities of neural networks can be further utilised in an attempt to develop an artificial conscience or awareness in the elevator control system. People usually use prior knowledge to solve familiar problems and do not necessarily do any new calculations for every required decision. Decisions made by the most efficient stand-alone elevator control system and the information that directed to that decisions need to be continuously stored in a database for each specific elevator controller. This case specific decisions can now be used as training and test data to build a neural network. The controller can then utilise the trained neural network to make quick and reliable decisions without the need to run a full computational routine for every change in the traffic compilation.

In order to ensure that the neural network controller still makes accurate and optimal decisions, it has to be evaluated and re-trained as required. By reflecting on its own decisions and to be able to know when to be re-trained can be seen as a form of cognitive awareness and reasoning. Thus the created neural network will represent a part the artificial conscience of the controller we are attempting to create.

7.5.2 NEURAL AUTOMATED REASONING HEAD: ONLINE LEARNING

Existing and new control philosophies can be easily tested and compared offline by any testing platform and by an elevator simulator, however we are introducing the possibility of an online testing platform. An online testing platform should be able to learn new control algorithm with the provided instruction set, respond cards and predefined conditions. Performance of the control algorithm is then to be tested and compared against the known algorithms. The notion of it to be an online exercise, means that the defining, testing and comparison stages occur automatically while the controller is online. This function of the controller will be known as the Neural Automated Reasoning Head (NARH). The NARH uses key performance indicators with previous score sheets for all trained algorithms to compare them to the newly learned algorithm.

The NARH should recognise the current traffic demand in the building and then be able to select the best online control algorithm to handle the identified demand. The chosen algorithm is then applied to the unprocessed requests together with the requests still in the system to produce the most optimal route. Thus we have two learning process for the newly proposed controller function.

First the control algorithms are tested by offline simulations as conducted in Chapter 8. The performance indicators, which are waiting time, power consumption and throughput maximisation, are stored for every Poisson passenger arrival rate across a time period. For every Poisson interval the best algorithm can be chosen from the collected data. The chosen algorithm will be the preferred algorithm for the specific passenger demand distributions it was tested against. However, to establish what the best solution is for any possible traffic distribution that can occur online, requires a Neural Network (NN).

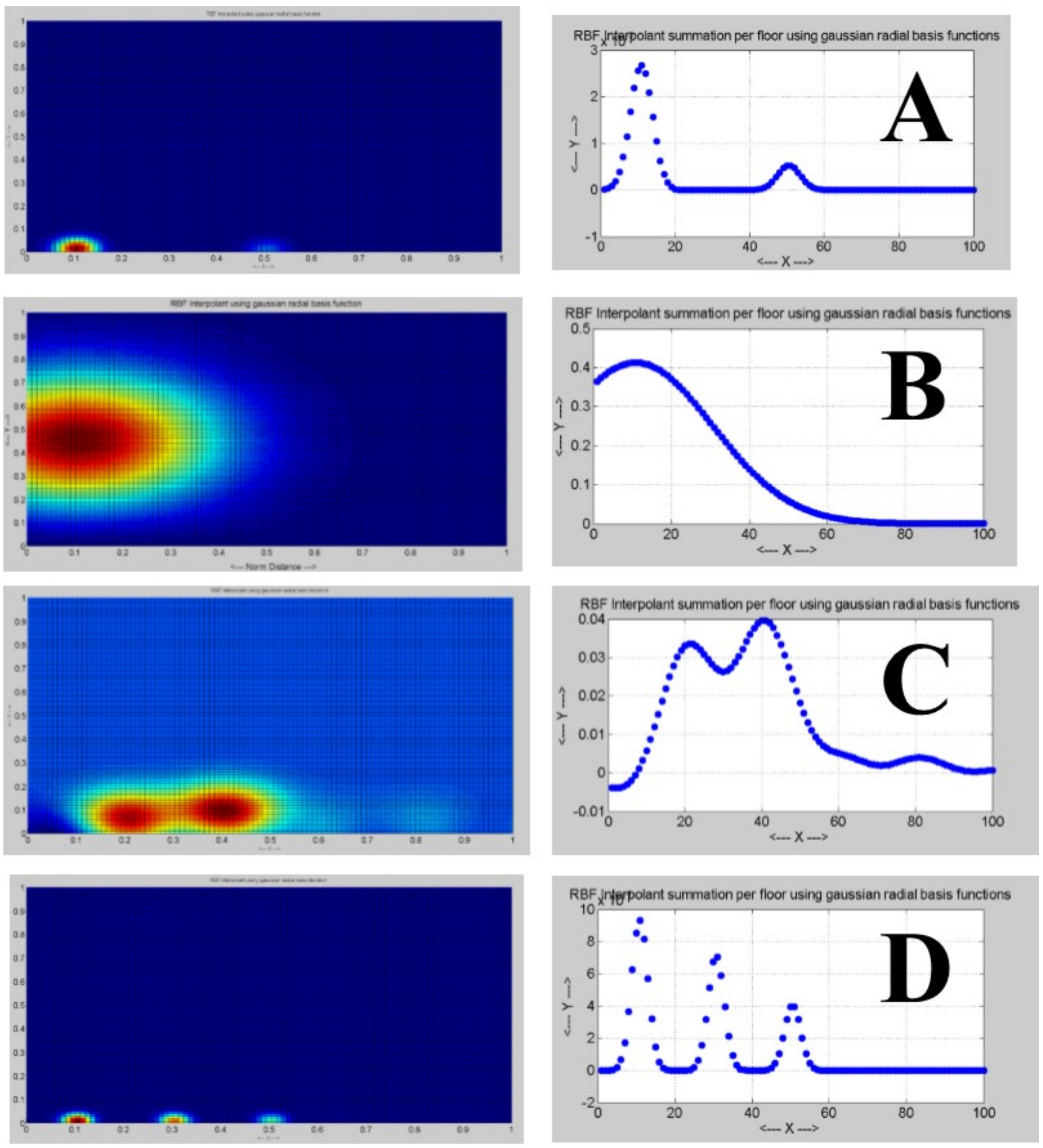


Figure 7.6: RBF Network Representing Building Passenger Demand for Different Time Periods.

The input neurons that are required for the NN are the existing or simulated traffic demand in the building. It can be either the amount of people or accumulated waiting time from all passengers that have not been serviced yet. The accuracy of the NN is improved if the input information is not independent data points like passenger arrivals, but rather from a conjoining network of data. The building passenger demand is modelled with Radial Base Functions (RBF) as discussed in Chapter 6 with either accumulated waiting time per floor or total amount of passengers waiting at each floor. The data points collected from each floor are then interpolated with a Gaussian radial base function integrated with an exponential test function provided by Equation 6.3.6. The result is a representation of the collected data points and also their influence on neighboring floors. This method is iterated for every Poisson passenger arrival rate period that is directly related to the simulated data that was used to obtain the best control algorithms described in the previous paragraph. Thus the input data for the NN is the RBF network for every Poisson interval and the output used for supervised training is the best chosen algorithm for the same interval.

Figure 7.6 illustrates different RBF networks for different time periods. The graphs on the left side are the continuous differential surface for a time period with an x-axis defined as normalised floors and the y-axis as the normalised interpolated data. The graphs on the right are the independent data points, either normalised waiting time or amount of passengers plotted against the normalised floors. The floors are normalised with their respective height or distance as a percentage of the building height. For example the ground floor is 0 and the top floor is 100% of the maximum height. Graph A was taken just as the up-peak traffic is arriving, graph B is in the middle of the up-peak traffic state, graph C is from a down-peak traffic state and graph D represents interfloor traffic.

7.5.3 SUMMARY

Each elevator group has generally one control algorithm implemented based on its building traffic environment and the company that installs it. The single control algorithm or chosen philosophy has a few response instructions to deal with the different traffic states, like up-peak, down-peak and interfloor movements, but it is preset and not adaptable to changing conditions. This section has introduced a more adaptable approach, where the controller is trained with multiple control algorithms and various simulated traffic conditions. The controller with the developed Neural Automated Reasoning model can now establish the best algorithm for every period throughout the day.

The main characteristics of the new approach can be summarised as follows:

- Controller is not limited to one control philosophy.
- The optimal control algorithm is automatically chosen and not pre-set by a designer or Expert System.
- Control responses are not triggered by a traffic state, but continuously changing with the accumulated data network (RBF).
- Online learning is a possibility.

7.6 INTELLIGENT ELEVATOR CONTROLLER SUMMARY AND CONCLUSIONS

The known software structure for computer control provided by Figure 2.1, was adapted into the Intelligent Elevator Controller (IEC) developed throughout this Thesis, presented by Figure 7.7. Firstly a Neural Elevator Model is developed throughout Section 7.2 with reference to Chapter 4. The Neural Elevator Model provides accurate power consumption feedback and running time predictions to be used in any cost function calculations, mostly for the TSP and VRP optimisation algorithms. It is also used to provide feedback to the user of the expected service time or car arrival time per floor.

Secondly the Expert System of pre-programmed control algorithms are loaded into the IEC, with compiled instruction sets, respond cards and conditions predefined. A few examples of possible control algorithms are discussed throughout Chapter 8, and are used to train the Neural Automated Reasoning Head (NARH) as discussed in Section 7.5. The Expert System is based on several years of Elevator development and proven control philosophies, and most are undisclosed by the Original Equipment Manufacturer (OEM). However any control algorithm can be fed into the system at any time, to ultimately improve the IEC and are then included with the general training data. With reference to Section 7.4 the provided control algorithms are further extended into individual Neural Smartstart networks with optimised configuration settings.

As new requests are received they are fed into the Dynamic Service Request Database (DSRD). The Neural Traffic Distribution Network (NTDN) reads the most recent requests from the database together with the newly unprocessed requests and provides a predictive traffic demand pattern. The Neural Automated Reasoning Head recognises the input traffic demand pattern and

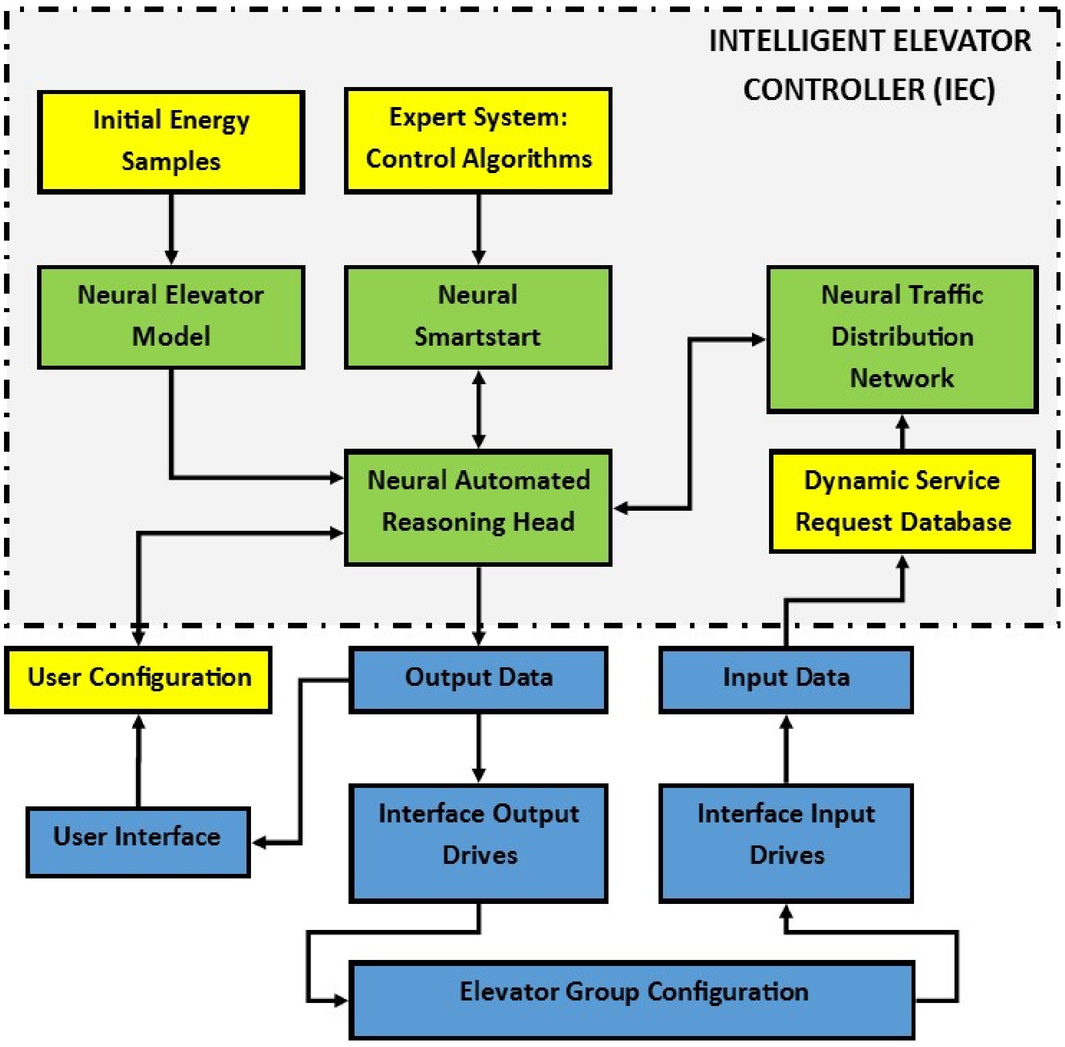

INTELLIGENT ELEVATOR CONTROLLER (IEC)
Initial Energy Samples
Expert System: Control Algorithms
Neural Elevator Model
Neural Smartstart
Neural Traffic Distribution Network
Neural Automated Reasoning Head
Dynamic Service Request Database
User Configuration
Output Data
Input Data
User Interface
Interface Output Drives
Interface Input Drives
Elevator Group Configuration

Figure 7.7: Intelligent Elevator Control Structure

selects the best online control algorithm that it has been trained with. The algorithm is then applied to the unprocessed requests together with the requests still in the system to produce the most optimal route as output data. Each car then decodes the output data through its own interface output drives and executes the instruction as required. The user also receives the output data either through the BMS or at each landing console.

To conclude, the developed neural online models for projected waiting times, probability networks and power consumption feedback were combined to form a new Intelligent Elevator Controller (IEC) structure as opposed to the Expert System approach, mostly used in present computer based elevator controllers. The research field of neural network science, was successfully utilised and the goal of creating an artificial intelligent (AI) controller was realised.

Chapter 8

ELEVATOR CONTROL SIMULATION

8.1 TRAFFIC GENERATION AND COLLECTION

In order to implement and benchmark the intelligent elevator model created in this research project, we require the input information from accurate traffic generation methods. A complete random generation method together with two Expert System traffic flow generation methods shall be developed in this Thesis to simulate deferent traffic patterns for comparison purposes.

8.1.1 COMPLETE RANDOMISATION

The first simulated traffic method is based on a complete traffic flow randomisation. The traffic generator class uses a randomise function namely `randInt()` as defined by Listing 8.1. It creates a pseudo-random number between the maximum and minimum input data from the `java.util.Random` library and implements a seed which is set to the time of execution in nanoseconds accuracy. The `nextInt()` method is exclusive of the top value, thus adding 1 is to make it inclusive. By using this method the time, number of passengers and floors can be randomly generated and stored into a database.

Listing 8.1: Complete Randomisation Function.

```java
public int randInt(int min, int max) {
   Random rand = new Random();
   rand.setSeed(System.nanoTime());
   int randomNum = rand.nextInt((max - min) + 1) + min;
return randomNum;}
```

However the randomise function limits the database to theoretical numbers and is not reproducing any kind of human behaviour or traffic patterns. In order to generate more representative traffic distributions we look to incorporate more commonly recognised elevator traffic patterns, namely an Expert System, instead of relying on complete random sample generation.

8.1.2 EXPERT SYSTEM: BIASED TRAFFIC PATTERNS

The second traffic flow generation method is based on a traffic flow benchmark, taken from the Elevator Traffic Handbook, written by Dr Gina Barney [11] and is illustrated in Figure 8.1. The traffic flow can be predominantly classified in the following groups: up peak, down peak, mid-day and interfloor traffic flows. The suggested passenger demand rate and the time of day when each traffic flow grouping normally occurs will definitely differ between different buildings, but the generalised curves can be adjusted as required. The intelligent elevator control model is written in such a way that each identified traffic flow pattern is set to an adjustable ratio of the up peak values. These ratios are adjusted after a pre-determined period to increase the accuracy of the probability variable against actual traffic flows. As the up peak values differ from day to day, the model can adjust its probabilities accordingly without necessarily changing the traffic model. The standard templates are also taken from Elevator Traffic Handbook for the traffic flow groupings and are utilised as the starting point for our simulation, where it can be adjusted throughout the program execution as required.

The adjustable variables will be the period in which the different patterns occur and the defined building population. From the initial up peak curves the following are estimated as being accurate: 53% of the population arrives within 62.5% of the overall up peak period and 27% arrives in the last 37.5% which effectively state that only 80% of the building population is occupying the building at one time [11]. The same goes for the initial down peak curve where period 3 is set to 1.6 times the up peak percentage population and almost linearly reduced until everybody is out of the building. The down-peak period is considerably shorter that the up peak curve, because of generalised human behaviour when it is time to leave. However this model does not register or store the destination floor information of passengers in the up-peak flow, so the down-peak flow is generated randomly from any floor. If the destination floors could have been recalled; the probability of the amount of passengers also leaving the building from that specific floors could have been increased. Interfloor traffic pattern movements are classified to be between 30% and 36%

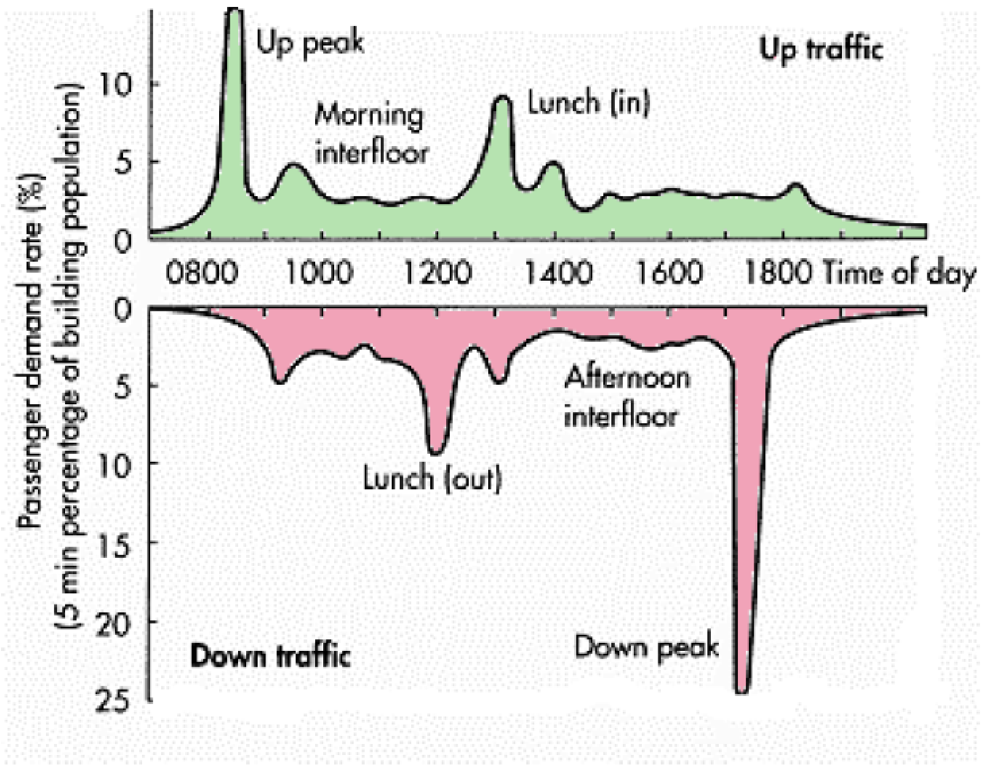


Figure 8.1: Passenger Demand Benchmark for a Typical Building [11].

of the building population in one hour [11]. To further define the interfloor traffic flow's; we divide 40% of these passengers going up, 40% going down, 10% from main floor up and from any floor to the main floor also to be 10% [11].

After the initial biased traffic patterns have been defined, the actual random arrival patterns can be established via the Poisson random generator as discussed in Chapter 5, Section 5.3.2. The 24 hour cycle can be generated in the same way the expected up-peak traffic pattern was created in Section 5.4.1 and extended to include the initial down-peak and interfloor traffic patterns as well.

8.1.3 EXPERT SYSTEM: PASSENGER PERSONALISATION EXTENSION

The third traffic generator method to be implemented is an attempt to increase the elevator's human interaction abilities in order to reduce the barrier between the elevator control system and its passengers. In order to assist in improving our probability curves and generation of real-world traffic patterns;

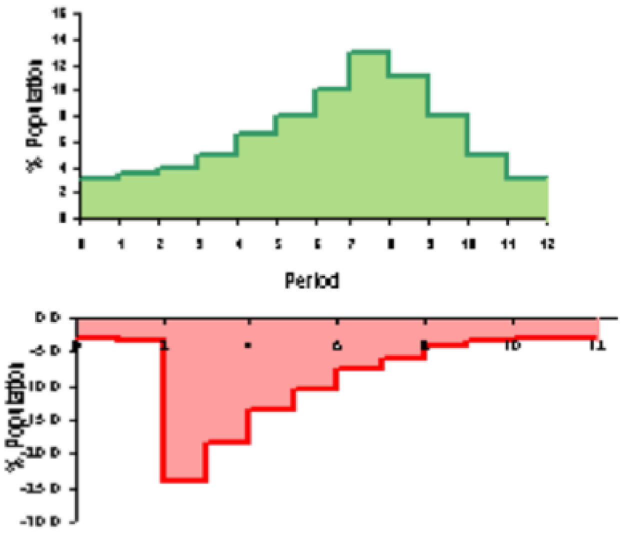


Figure 8.2: % Population vs. Period for a Typical Building [11].

more information is required about each passenger. Each passenger is classified into groups namely either a visitor or a member of the company. Each personnel has a name, a permanent workspace in the building and a work schedule that is normally followed. A visitor to the building can also provide information about the specific floors he or she will be visiting and at what time. Some companies are implementing the integrated access control (IAC) system to their respective premises. This system works with a RFID tag, where permission must be granted for any person who enters the building and various rooms or areas inside the building and logs the timestamp of every activated door. This system can be utilised further by placing these RFID receivers at each elevator landing floor to register the information about the person as in practise by some elevator companies. The information is then to be used in actual pattern classification and compared to the simulated values.

Listing 8.2: Passenger Information Class Definitions.

```
public class Personnel {
  protected String NAME;
  protected String UNIQUE_NUMBER;
  protected int[] work_start_time;
  protected int[] work_end_time;
  protected Floor work_station;
}

public class Visitor{
  protected String NAME;
  protected String ID_NUMBER;
  protected int[] visit_start_time;
  protected int[] visit_end_time;
  protected Floor visit_station;
}
```

8.2 SINGLE ELEVATOR CONTROL SIMULATION

8.2.1 GENERAL BASELINE TRAFFIC CONTROL

The baseline main method starts with utilising the present time and the moving direction. Then the processor looks for the next floor to service from the `next_floor_method()`. When the floor to service next is found, the controller dispatches an elevator to that floor and adds a delay to the overall time. The variable delay or non-negative routing cost is estimated by Listing 8.3 or obtained through the neural network energy model, provided in Section 7.2.

Listing 8.3: Routing Time Estimation for Unit 2 Aux Bay.

```
private final int doors_opening = 4.9*1000
private final int doors_closing= 4.9*1000
private int people_moving_in_out= people_count*1000

  elevator_moving_time = distance_travelled /rated_speed;
  total_delay = doors_opening + doors_closing + ↩
     people_moving_in_out;
```

To identify the next floor to service, the following logic is used to write the `next_floor_method()`:

- Create and initialise passenger lists for each floor and each car separately.
- Register all floors where the directional buttons have been pressed.
- Register if the up or down button was pressed at the landing floor.
- Add all passengers to their respective lists.
- Register all destinations where the destination buttons have been pressed by people in the elevator car.
- Add passengers in car to the passenger destination list.

If the elevator is going in the up direction, identify the landing floors from the current position to the top floor. Service the closest floor that satisfies the following criteria: somebody pressed the up button or has reached this/her destination floor. If none of the identified floors were serviced then add the following criteria: if any directional buttons were pressed or if the car is not empty, service the closest floor that satisfy the criteria from the current position to the ground floor, or vice versa if the down directional button was pressed. When a floor is serviced, all passengers in the elevator car that has reached their destinations are removed from the car passenger list and all passengers waiting at that respective landing floor is added. After all passengers were serviced, the waiting times at each landing and from the cars are then calculated and documented to compare later with other traffic control philosophies.

8.2.2 TRAVELING SALESMAN PROBLEM IMPLEMENTATION

8.2.2.1 GENERAL TRAVELING SALESMAN PROBLEM THROUGH BRUTE FORCE

The general traveling salesman problem (TSP) is established through a Brute Force (BF) programming philosophy; where every possible combination are generated and stored in a list or an array. Every floor can only be visited once in a planned route by each elevator car and does not have any time window constraints, however the number of elevator cars can be adjusted and each car does have load capacity limitation. We can develop a simulation model trough Brute Force as illustrated by Figure 8.3. With the `initialise()` method the `Traveling Salesman` class Object is created which travels between 8 floors for example, that result to 40 320 different iterations, where the combination generator method checks that no floors are repeated in a floor order sequence. Once the list of all combinations has been established; passenger samples are collected through a database of Poisson generated randomised samples based on a 24 hour Expert System cycle. These passenger samples are then released to the controller in a so called real-time simulation to create an online routing problem, where future information is not available until it has come to past.

The two deterministic online transportation problem strategies, Ignore and Replan, as mentioned in Section 2.2 have been implemented in this version of the model. Where the Ignore strategy will continue with the planned route, neglecting any new passenger requests and the Replan strategy would re-calculate and provide the optimum route for every new request. Via the `get_floor_order_BF()` function these two strategies are optimally combined to reduce the total waiting times experienced by the passengers. When the method requires for a Replan strategy to be executed, all combinations are

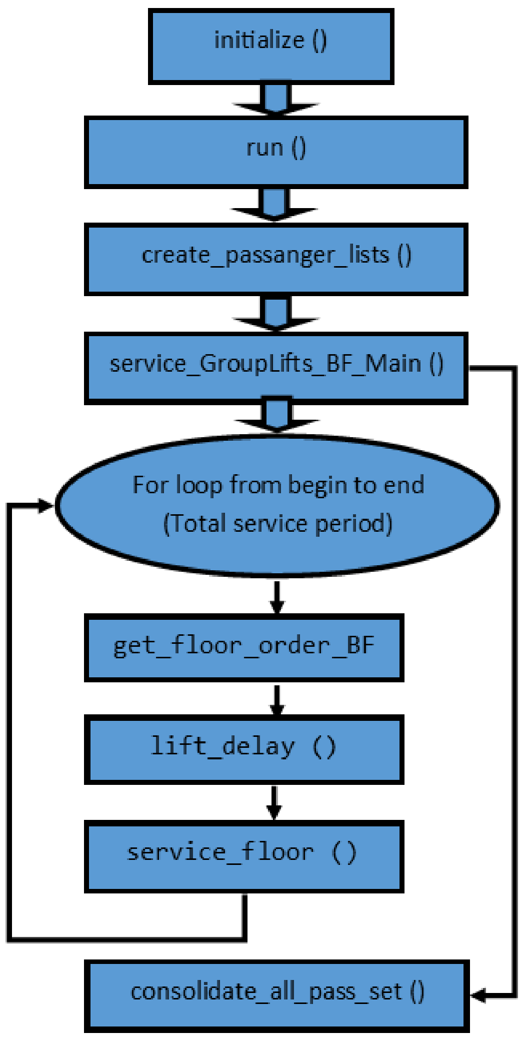

initialize ()
run ()
create_passanger_lists ()
service_GroupLifts_BF_Main ()
For loop from begin to end
(Total service period)
get_floor_order_BF
lift_delay ()
service_floor ()
consolidate_all_pass_set ()

Figure 8.3: General Traveling Salesman Problem Simulation Trough Brute Force.

tested to obtain the best result (waiting times) by the cost function as described in Section 6.2 and specifically Equation: 6.2.5. The best floor sequence becomes the most recent global planned route, where the elevator car then proceeds to service the next planned floor after the simulated delay was added to the service time as required. The procedure to establish which strategy to implement after each new request; is done via alternating the allowed capacity of each elevator, meaning if the elevator car has reached a certain capacity no alterations would be conducted to the planned route. The procedure is discussed in Section 7.4.

8.2.2.2 TSP WITH GENETIC ALGORITHM PROGRAMMING

In the first attempt to establish an intelligent traffic control program; a genetic algorithm is implemented. The traveling salesman class is initialised with the following variables and steps which are an adaption from Heaton's earlier work [3].

Table 8.1: The Traveling Salesman GA Class Initialisation.

Variable	Value
int FLOOR_COUNT	4
int POPULATION_SIZE	50
double MUTATION_PERCENT	0.10
int MATING_POPULATION_SIZE	POPULATION_SIZE/2
int FAVORED_POPULATION_SIZE	MATING_POPULATION_SIZE/2
int CUT_LENGTH	FLOOR_COUNT/5
int generation;	0

Step 1: Create the initial chromosomes up to the amount of the pre-defined `POPULATION_SIZE` variable, where each chromosome is a different order of the pre-set number of floors. Set the cut length which determines how much genetic material to take from each “partner” when mating occurs. Set the mutation percentage; meaning the percentage of the offspring that will be mutated or the probability that a mutation will occur.

Listing 8.4: Chromosome Decleration.

```
chromosomes[i] = new Chromosome(floors,current_floor);
chromosomes[i].setCut(cutLength);
chromosomes[i].setMutation(TravelingSalesman.↩
    MUTATION_PERCENT);
```

Step 2: Calculate the fitness of each chromosome by Listing 8.6. The fitness or the cost of each chromosome is the summation of each Pythagorean distance from one floor to the next in the current chromosome defined floor order list, starting from the first floor to the last planned floor to be visited. The cost of a chromosome determines its rank in terms of being able to mate and not to be killed off.

Listing 8.5: Chromosome Fitness Calculation.

```
for ( int i=0;i<cityList.length-1;i++ ) {
   double dist = floors[cityList[i]].proximity_xyz(floors[↩
      cityList[i+1]]);
   cost += dist;
}
```

Listing 8.6: Proximity Function.

```
public int proximity_xy(int x, int y) {
  int xdiff = xpos - x;
  int ydiff = ypos - y;
return(int)Math.sqrt( xdiff*xdiff + ydiff*ydiff );}
```

Step 3: Sort or rank all chromosomes in ascending order based on their fitness calculations. The first set of chromosomes will be classified as generation zero and so on. The chromosomes are sorted by their cost, by method: `Chromosome.sortChromosomes(chromosomes,POPULATION_SIZE)`

Step 4: Mate the chromosomes in the favoured population with all the other chromosomes in the mating population. In this instance the favoured population are a quarter of the total population. The favoured population can be seen as the mother chromosomes and the father chromosomes as any random chromosome between zero and half the total population size. The mating function is provided by Listing 8.7 [3].

Listing 8.7: Chromosome Mating Call Method.

```
for ( int i=0;i<favoredPopulationSize;i++ ) {
  Chromosome cmother = chromosomes[i];
  int father = (int) ( 0.999999*Math.random()*(double)↩
     matingPopulationSize);
  Chromosome cfather = chromosomes[father];
  mutated += cmother.mate(cfather,chromosomes[ioffset],↩
     chromosomes[ioffset+1]);
  ioffset += 2;}
```

The mating function is implemented where the favoured “mother” chromosome mates with a random “father” chromosome and then returns the amount of mutation that was applied. First a cut range is established between cut point 1 and 2. Cut point 1 is defined as any floor between zero and 80% of the total possible floors and cut point 2 is defined as cut point 1 plus the cut length. All the mother genes are copied over to offspring 1 except the genes in the cut range witch belongs to the father chromosome and the same with offspring 2, which get all the genes of the father chromosome except the genes in the cut range which belongs to the mother. For a small probability defined with `Math.random() < mutationPercent`, mutation can be set to occur; where random genes are swapped between the 2 sets of offspring. The number of mutated chromosomes are counted and later used to calculate the mutation rate of the mating population as a whole. A new generation was created as result of this step and copied over to the second half of the total population.

Step 5: The new generation is now moved from the second half of the population to the mating population where the fitness of every chromosome is calculated again as in step 2 and 3. The mating population is now sorted based on the fitness calculations. Out of this generation the best chromosome can be found which represent the optimal TSP route for generation zero

This process is iterated, until either the required minimum cost between floors is reached from the best chromosome or when the best chromosome is consecutively found a number of times. The TSP was implemented symmetrically at first, meaning that it the distance from city x to city y is the same as from y to x, but it resulted in a system which is not conducted heuristically and is very static in a problem solving perspective. The programming structure does not allow for floor variables to change once the process has started, which means that ever-changing variables are not dynamically handled through-out. For instance simulated delays and waiting times as a result of the floor order are not dynamically accounted for. To account for more spatial distance calculations; an asymmetric TSP is established. Other less than optimal qualities or shortcomings from the TSP were identified after the TSP was implemented with GA but without neural networks, for example:

- The TSP is structured in such a way that the floor order is provided by looking at the overall picture or the overall shortest distance to travel between floors. Thus if the whole floor order is not carried out as calculated, the result will be less than optimal. The conclusion is that this technique is not suited to provide only single floor recommendations. Also the general TSP only looks at visiting each city ones in a trip, but extended as required.

- The system requires a fully defined set of variables, where any changes in the definition can have a major impact on the result. For example the waiting time variable can either be defined in terms of average waiting time per floor or per person, maximum time a person has waited per floor or the summated waiting time per floor. In this instance the program requires Expert Knowledge in order to define the different variables in order to attempt a balance between the individual and the group requirements. This system can then rather be defined as an Expert System and as a result; loses its genetic capabilities as originally intended.

- During normal elevator traffic flow's, excluding high midday, morning and afternoon peeks, a lot of floors have no traffic registered and don't need to form part of the floor order calculation. This result in an often simplistic TSP and can be often solved by only a few iterations. The capabilities of genetic algorithms without an ability to evolve are in reality not utilised and not necessary in its current form.

- In the current TSP structure it will fill up the elevator car towards its full capacity as the elevator car moves from floor to floor, but there is potential of finding a more optimal solution of filling up the car towards its full capacity without overlooking other floors until the car has space again.

8.2.2.3 TSP WITH SIMULATED ANNEALING PROGRAMMING

In an attempt to improve the traveling salesman problem, we will be implementing the simulated annealing process via `TSP_SimulatedAnnealing.class` to get the best path.

Step 1 is to create a random path for the traveling salesman by initialising a random floor order. With the following parameters:

Listing 8.8: TSP Simulated Annealing Parameters.

```
double START_TEMPERATURE = 10;
double STOP_TEMPERATURE = 2;
int CYCLES = 100;
```

Step 2 is to run the `iterate()` method until the cost calculated by the best retained path is not improved over the amount of iterations set out by the `CountSame` variable. During each `iterate()` method the floor order is likely to be changed slightly with the `randomise()` method according to the freedom set out by the simulated annealing process. The simulated process is repeated by the amount of cycles defined, where the floor order is potentially changed with each cycle. The freedom of the process is constrained to the start and stop temperature ratio. The higher the starting temperature the more likely is a larger change in floor orders from one iteration to the next.

After a few attempts of implementing this method the following can be concluded with a measure of uncertainty:

- It seems that simulated annealing can only be implemented if the cost calculations are between non-spatial variables. The process compares different sections of each path against each other in an attempt to reduce the cost between cities, but it does not focus on improving the overall cost of the entire path. Thus for example when variables like waiting times are closely compared from one city to another; the traveling salesman will visit them in quick succession, because they have similar waiting times, but will not however visit them in the most optimal order to have reduced overall waiting times.

- Additional research is required in order to prove that this method can be used to improve the TSP for the elevator traffic control application.

- It does not seem that the first city in the path will ever be changed by the simulated annealing process. The iteration method only attempt to draw all the cities closer to the first city in the path order; in an attempt to reduce cost.

- A possibility to reduce the waiting time can be accomplished by comparing different paths like in the case with GA, but without losing the basic principles of simulated annealing. The basic idea of having various levels of randomness and freedom to optimise the floor order should be retained.

8.2.3 ALGORITHMIC PERFORMANCE RESULTS

As mentioned before we are attempting to make the overall system less dynamic, in order have an actual system response that is closer to the optimal and theoretical calculations. By means of a competitive ratio, we can have an indication of how the various probability methods have decreased the system' dynamism. The competitive ratio Cr_A for an algorithm A can be defined for an instance I [28]:

$$Cr_A = sup\frac{z(A,I)}{{}^{*}z(I)} \tag{8.2.1}$$

Where $z(A,I)$ is the cost of the solution by the algorithm A and ${}^{*}z(I)$ the cost of the optimal solution if all data of the instance I were available beforehand. The dynamic system can also be described as an online system where the input arrives in different phases throughout the process. An offline system is controlled with optimal offline algorithms, where all the required input data are available from the first instance of the process. The competitive ratio can thus be defined as the supremum cost of the online algorithm over the cost of the off-line algorithm.

For the lower bounds of a probabilistic algorithm; we can calculate the competitive ratio $\overline{c}$, with Yao's principle:

$$E_Y[ALG_y(\sigma_x)] \geq \overline{c} \cdot OPT(\sigma_x) \tag{8.2.2}$$

Where $ALG_y : y \in Y$ are a set of the applicable deterministic online floor order sequences as per strategy, see Section 2.2, and $\sigma_x : x \in X$ is a set of possible requests sequences [16].

The expected cost of the deterministic online sequences with respect to a probability distribution X is defined as [16]:

$$E_Y[ALG(\sigma_x)] = \int_x ALG(\sigma_x)dX \tag{8.2.3}$$

The algorithmic performance results can be divided between the approximation performance and the competitive performance of the different strategies and techniques, as describes throughout this thesis. The different machine learning techniques are referenced by Table 8.2. We can calculate the approximation ratio ρ as defined by [16].

$$ALG(I) \leq p \cdot OPT(I) \tag{8.2.4}$$

Where $ALG(I)$ is a deterministic online floor order sequence, see Section 2.2, and $OPT(I)$ is the optimal sequence if all requests were known before the instance initiated.

Table 8.2: Algorithmic Performance Results for Single Elevator Control.

Machine Learning Techniques	Approximation Ratio	Competitive Ratio
Baseline	2.02	3.55
Brute Force	1.13	2.172
Genetic Algorithms	1.14	2.185

According to Krumke [16] , the Smartstart strategy is the best possible online algorithm with a competitive ratio of 2, the BF and the GA algorithms implemented in this section compares well with the stated benchmark for a single elevator application. The baseline control being implemented by the present configuration, were proven to be the least competitive with a competitive ratio of 3.55 and an approximation ratio of double that of the best possible solution.

8.2.4 TRAVELING SALESMAN PROBLEM IMPLEMENTATION RESULTS

In the first set of results a baseline was established to be used as a reference point for the various elevator controller performance indicators. The baseline refer to the basic controller decision making process that are still found in may elevator installations, where passengers get serviced if their destination floor is in the same direction as the traveling elevator car. With the baseline there are no implementation of any destination control, thus the controller only reacts to directional buttons activated at the landings and from inside the car.

The traveling salesmen problem was simulated with the use of Brute Force and Genetic Algorithm techniques and the key performance indicators were throughput maximisation and waiting time reductions. The implementation of the TSP resulted in a significantly improvement on these 2 areas and was proven to be superior against the baseline. The average waiting time spend at the landings and in the car has improved by 30% with the Brute Force TSP and 33% with Genetic Algorithm TSP. Time spend in the elevator car has not improved against the baseline, because the TSP calculate overall waiting time cost of all waiting passengers. This leads to higher waiting times in the car, but lower waiting times experienced at the landings.

In Figures 8.5 and 8.6, it can be seen that the TSP handles peak traffic periods more effectively than the baseline. The maximum accumulated amount of passengers at one instance reached 70, where the BF TSP reduces this number by 23%. When the maximum amount of people waiting at a time is divided by the number of floors in the elevator configuration the figures becomes more acceptable, however when these figures are realised at only a few floors it becomes problematic. For instance during up-peak traffic periods; it becomes clear that additional elevator cars should be introduced into the elevator configuration to be able to handle the building population. But for the purpose of testing the different controller strategies, the worst case traffic patterns can illustrate how each perform and where weaknesses are identified.

Table 8.3: Passenger Average Waiting Time Results for a Worst Case Scenario.

Average Waiting Time	Baseline	TSP BF	TSP GA.
In Car	2m 59s	2m 29s	2m 37s
At Landings	10m 54s	7m 39s	7m 19s
Total	13m 53s	10m 8s	9m 56s

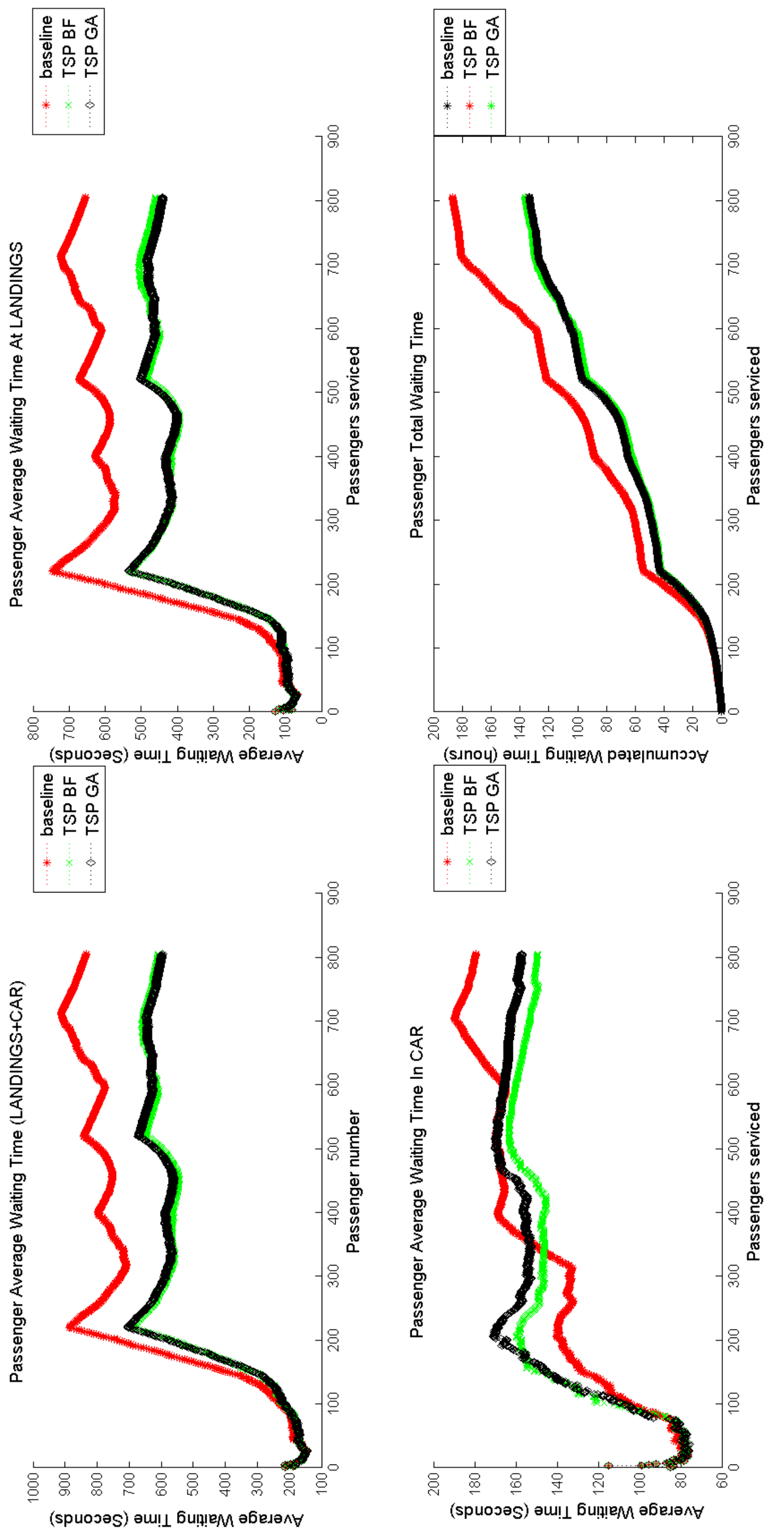


Figure 8.4: Passenger Waiting Times for a Worst Case Scenario: Baseline vs. TSP BF vs. TSP GA.

The best results provided by Table 8.3 for the BF TSP and GA TSP were by implementing the Ignore on-line strategy when the available capacity in the elevator car has reached 54% of the rated load capacity. Which means that the planned route will be carried out regardless of incoming requests if there are more than 6 people in the car. When more capacity becomes available the optimal route is recalculated by the Replan on-line strategy and becomes the newly established planned route for the TSP. The number of estimated iterations initiated by Replan reached just under 24 million for the BF TSP and about 7 million for the GA TSP. As a result of the use of machine learning techniques like Genetic Algorithms; the overall number of iterations and computational time have been reduced considerably and have not effected the performance negatively with comparison to the Brute Force approach.

Table 8.4: Elevator Controller Service Performance Results.

	Baseline	TSP BF	TSP GA
Number of iterations	<1k	<24 000 k	<7 000 k
Compilation time	8s	5m 28s	1m 41s
Number of services	746	708	706
Maximum waiting passengers	70	54	57

8.2.5 POTENTIAL SAVINGS: AN ACTUAL CASE STUDY

A trip counter was installed at Aux Bay Unit 1 elevator at Majuba Power Station; in order to correlate the expected traffic density to the actual trip data being measured. Illustrated by Figure: 8.7 the mid-day traffic period for 11 Aug 2014 was recorded, which represented the lunch hour period for the building's personnel. The recorded data summated to a power consumption of 11.33 kilo watt-hour and 40 mega Joules. The elevator trip activity pattern correlates strongly with the Poisson arrival distribution as expected.

Majuba Power Station has 6 similar single Aux Bay elevators, one in each generating unit. With the general estimated personnel distribution and the expected visitor rates for the station, the power consumption from these 6 elevators can be extracted from our simulation model based on the installed control mechanisms and compared to the proposed TSP control philosophy. Table 8.5 provides the developed Energy Model results for a 5 day period with the Aux Bay U1 elevator configuration, divided into 3 different single elevator control approaches. As a result of realistic population numbers, in contrast to the worst case scenario tested previously, the performance improvements are

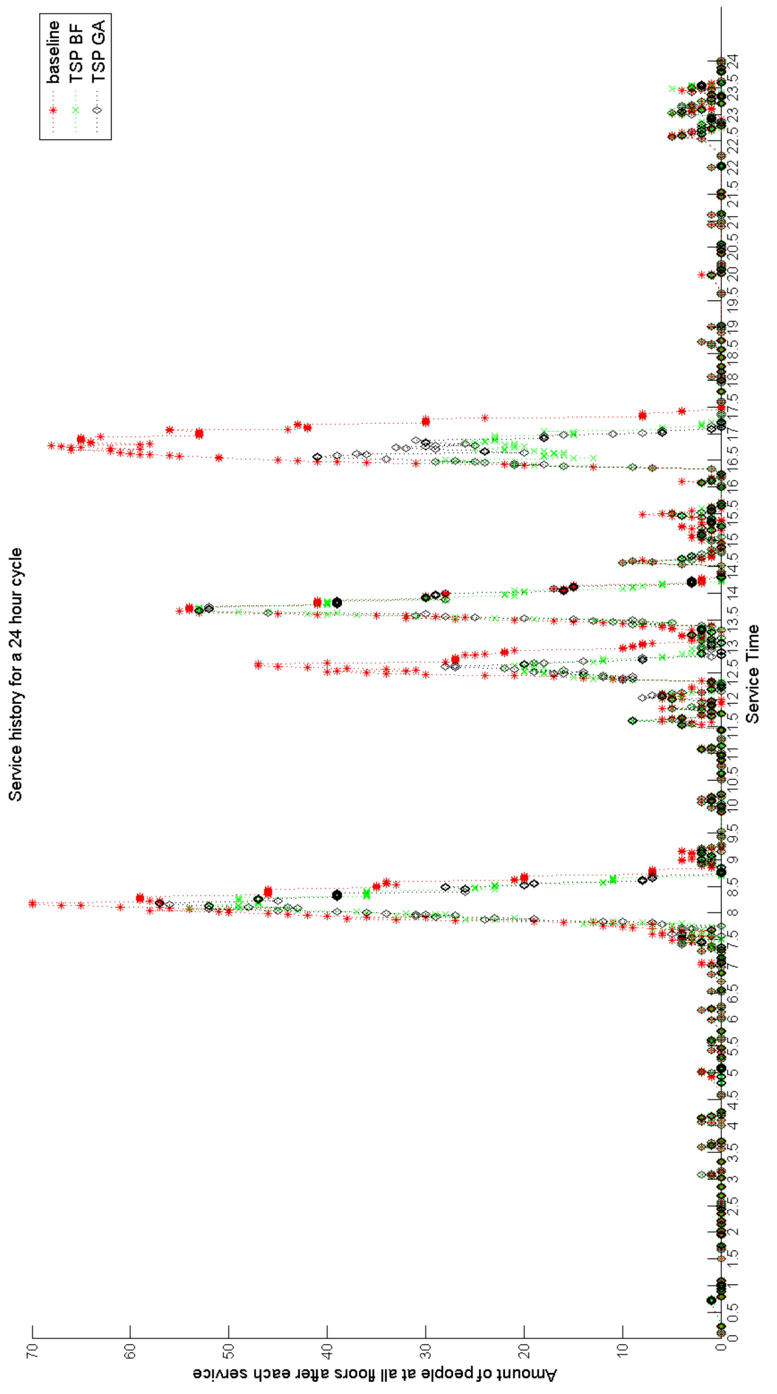


Figure 8.5: Elevator Controller Service Performance (with Service Time) for a Worst Case Scenario: Baseline vs. TSP BF vs. TSP GA.

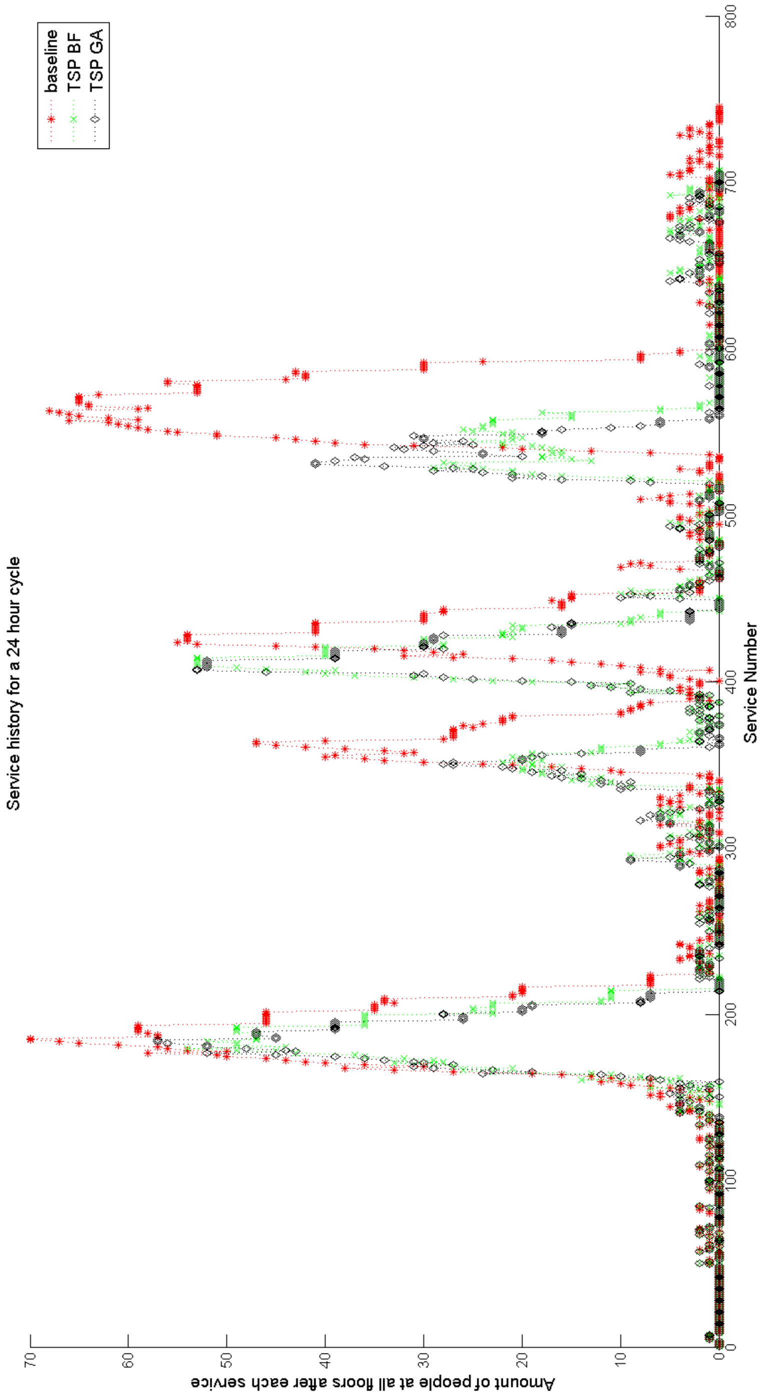


Figure 8.6: Elevator Controller Service Performance (with Service Count) for a Worst Case Scenario: Baseline vs. TSP BF vs. TSP GA.

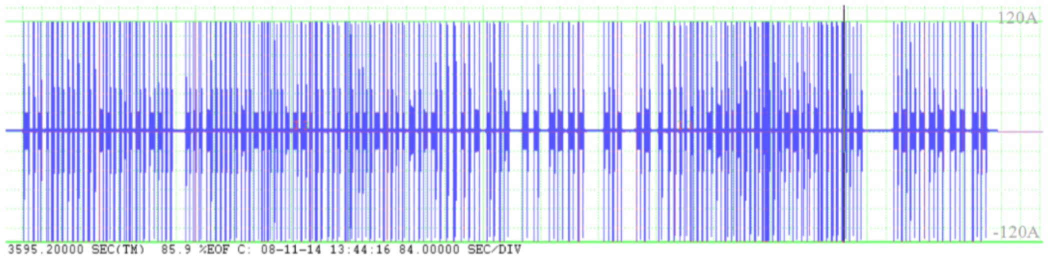


Figure 8.7: Actual Trip counter for the lunch hour period (Aux Bay U1, 11 Aug 14)

not as high, but are still noticeable. The BF TSP and the GA TSP performed relatively the same, because a 4 floor route only produces 24 cost iterations per Replan strategy. The TSP solutions improved the waiting times at the landings by more than 24% and the total waiting time by 20% over the actual installation results. However the power consumption over the 5 days were not reduced by the TSP solutions, because waiting time had priority over energy savings in this instance. It can be concluded that as a result of few floors and the close proximity of the floors that any potential savings are neglectable from the baseline case, however waiting times can still be improved.

The results from this simulation, also provides an indication of the usage of the current elevators. For a 5 day period the elevator car was stationary for 86.3% of the time, with total running time reaching 16 hours and 28 minutes. A total accumulated distance reached 46 km with a total power consumption of 277 kilowatt-hour (kWh), which translates to an expenditure of R 408 with R1.47 per kWh. With 6 similar elevators the projected expenditure for a year is around R120 000, with 80 MWh, 1.3 million stops and 13 000 km expected distance travelled.In Section 4.4 it was mentioned that if the elevator motor had a different configuration; where the 18 pole windings were not configured in the opposite direction to the 4 pole winding configuration and the drive had regenerative capabilities: The machine would have been operating in the generating state and performed regenerative braking, where a negative torque decelerates the motor, with the surplus kinetic energy reverted back through the regenerative drive until the motor comes to a stationary position. The actual minimum savings per trip was measured at 10.86% from 20 to 0m and at a maximum of 65.12% from 16m to 20m, which measured at 31.79kJ and 140.74kJ respectively. However to get a more representative savings percentage, it has to be measured in context over a period of time. It was established by the Energy Model that over the 5 day period; the 18 pole windings set consumed 278 MJ from the total consumption of 985MJ. Thus it can be stated that on average; 28.2% of the total energy consumed can be saved, as well as the energy that can be reverted back into the system, to declare additional savings.

This concludes that by upgrading the control system we can improve overall waiting time performance by 20% and by upgrading the actual elevator installation, the potential energy savings are more than 28% on average excluding the additional energy that can be reverted back into the system as well.

Table 8.5: Energy Model Simulation Results for a 5 Day Period with Aux Bay U1 Elevator Configuration.

Total	Baseline	TSP BF	TSP GA.
Stops	4748	4745	4727
Current	28.928 A_{rms}	28.558 A_{rms}	28.523 A_{rms}
Running Duration	16h 29m	16h 49m	16h 47m
Power Consumption	277.517 kWh	280.182 kWh	279.367 kWh
Energy	985.869 MJ	996.625 MJ	993.506 MJ
Distance travelled	46 290 m	47 626 m	47 586 m
Expenditure	R 407.95	R 411.868	R 410.669
Car Total WT	1d 22h 7m	1d 15h 42m	1d 16h 11m
Floor Total WT	3d 12h 13m	2d 15h 39m	2d 15h 43m
Total WT	5d 10h 20m	4d 7h 21m	4d 7h 54m
Avg. Car WT	0m 39s	0m 34s	0m 34s
Avg. Floor WT	1m 12s	0m 54s	0m 54s
Avg. Total WT	1m 52s	1m 29ss	1m 29s

8.3 ELEVATOR GROUP CONTROL SIMULATION - WITHOUT DESTINATION DISPATCHING

8.3.1 INTRODUCTION

In medium to high rise buildings, it is often required to have more than one elevator available to handle the additional passenger demand requirements. A multiple elevator configuration introduces the terms known as group supervisory control; where the controller is pre-programmed or pre-configured with a specific control philosophy. Various group control techniques are in practice today by well-known elevator manufactures, for example the Duplex/Triplex system (THV), a fixed sectoring priority timed system (FS4), a fixed sectoring common sector system (FS0), a dynamic sectoring system (DS), etc. [1]. These techniques have been studied and proven to be successful for the technology that was available at the time. This Thesis does not cover these techniques in detail, but would rather introduce new machine learning techniques to reinforce the ideas behind them. The main principle behind an artificial intelligent elevator is to avoid a set of structured conditions and pre-programmed response classes. Therefore a more appropriate term than group supervisory control is further used in this chapter, namely elevator car dispatching. Where the controller is a multi-core processor and dispatches elevator cars based on optimal route calculations and /or pre-programmed philosophies.

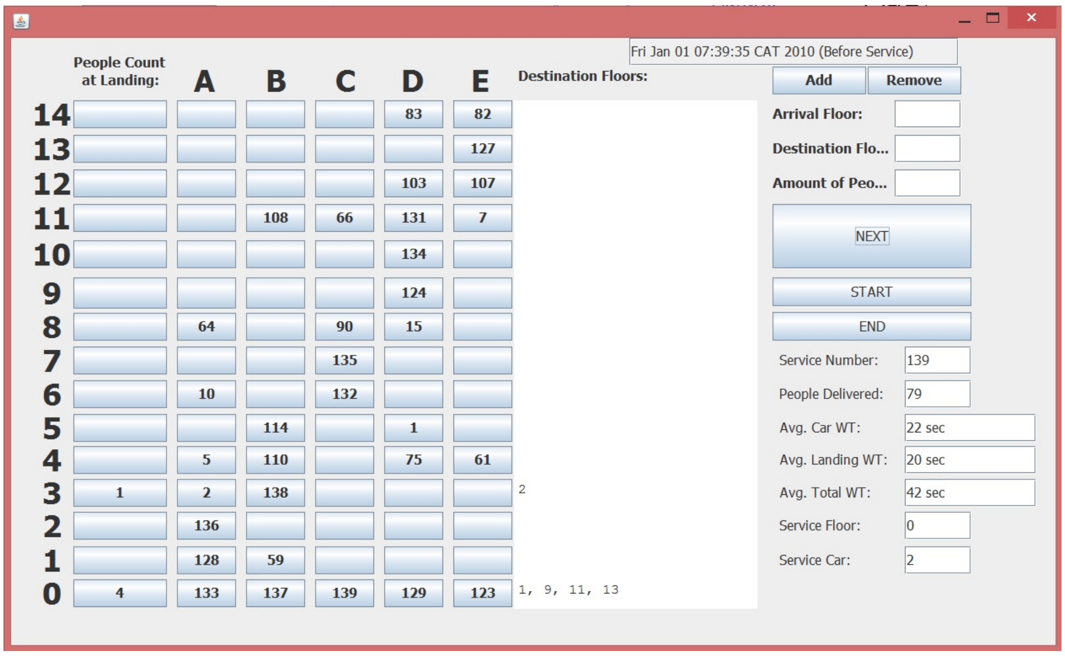
8.3.2 ELEVATOR CAR DISPATCHING

Elevator car dispatching are divided into two technology based control mechanisms. The first is where the passenger's destination floor is either known before the person enters the elevator car or after, but the designated car is not indicated to the passenger at any time. Meaning when the car doors open; it collects the waiting passengers and is not passenger specific. The second control mechanism is passenger-biased; where a specific car is allocated to each passenger. This mechanism is referred to as destination dispatching (DD). A benefit to the first control mechanism is that route optimising is continuous and does not have to be defined at the exact moment the passenger arrives. However the controller has a more complex task of optimising a route for all waiting passengers, rather than for individuals which are proven to require less computational resources.

An Intelligent Elevator Controller (IEC) Graphical User Interface (GUI) are presented by Figure: 8.8. The GUI is used as a graphical addition or resource to represent different elevator car dispatching algorithms with the same elevator configuration, namely a 15 floor building with 5 elevator cars. For the purpose of creating a controlled testing environment, the floor landings are set at; $\{0,\ 4,\ 8,\ 12,\ 16,\ 20\ 24,\ 28,\ 32,\ 36,\ 40,\ 44,\ 48,\ 52,\ 56\}$ meters and the elevator cars have a rated speed of 2.5 $m.s^{-1}$ and the building population can be changed as required.

8.3.2.1 VEHICLE ROOTING PROBLEM IMPLEMENTATION

The vehicle rooting problem (VRP) is an extension from the traveling salesman problem and has been introduced in Section: 3.4. When implementing a Brute Force (BF) VRP algorithm; every permutation are considered between the number of cars and the number of floors. Each valid combination are generated and with the use of a cost function either rejected or implemented as the optimal combination for the BF VRP dispatching algorithm. As an example; let's say that the controller has received passenger requests from the following landing floors; $0,\ 4,\ 9,\ 14$, where these landing floors can be added to a single elevator or divided between a maximum of 5 elevator cars. The total number



Fri Jan 01 07:39:35 CAT 2010 (Before Service)

	People Count at Landing:	A	B	C	D	E
14					83	82
13						127
12					103	107
11			108	66	131	7
10					134	
9					124	
8		64		90	15	
7				135		
6		10		132		
5			114		1	
4		5	110		75	61
3	1	2	138			
2		136				
1		128	59			
0	4	133	137	139	129	123

Destination Floors:

```
2

1, 9, 11, 13
```

Arrival Floor:	
Destination Flo...	
Amount of Peo...	
Service Number:	139
People Delivered:	79
Avg. Car WT:	22 sec
Avg. Landing WT:	20 sec
Avg. Total WT:	42 sec
Service Floor:	0
Service Car:	2

Figure 8.8: Intelligent Elevator Controller Graphical User Interface (version 1.0): Medium Rise Building Configuration.

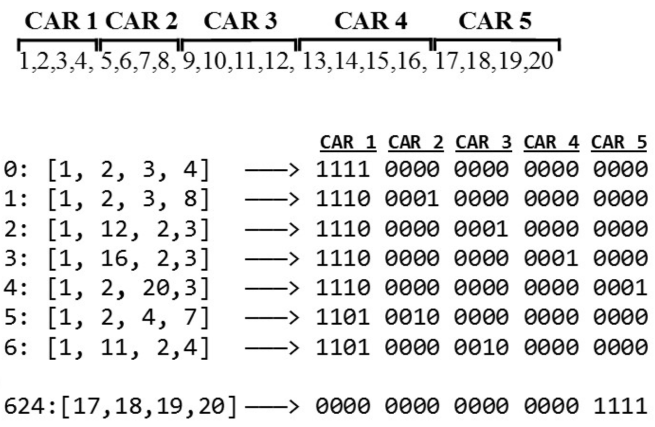
of possible false combinations can be defined with Equation: 8.3.1 to be 4 845 in this case.

$$\text{possible combinations} = \binom{\text{landings} \cdot \text{cars}}{\text{landings}} \tag{8.3.1}$$

However the combination count from Equation: 8.3.1 includes true combination duplicates, meaning the landing floors are presented to the cars multiple times just in different order combinations. However the optimal route for each car will be calculated later by the TSP algorithm, when new requests are added to the planned routes. Another constraint is defined which states that when a car has been allocated to a floor, then the other cars should not visit the same floor. To remove the duplicates and to avoid the same landing floors from being visited twice, the following method is introduced; false VRP numbers are generated between 1 and the amount of landings multiplied the amount of cars, see Figure 8.9. The elevator cars are presented with combinations of false numbers, which are also presented as binary generated numbers. These numbers are then converted to the exact floor numbers originally provided with `get_real_order_combinations()` method, provided by Listing 8.9. The total combination count have now been reduced from 4 845 to 625 for the permutation possibilities between 4 floors and 5 cars. Table 8.6 illustrates how quickly the number of required computations increase with additional landing floors to visit, even as little as 6 floors are proven to be a challenge.

Table 8.6: BF VRP: Number of Calculated Steps for a 5 Car Configuration.

Landings to visit	False Combinations	True Combinations
1	5	5
2	45	25
3	455	125
4	4845	625
5	53130	3125
6	593 775	15625
7	6 724 520	78 125
8	76 904 685	390 625
...	...	...
15	2 280 012 686 716 080	30 517 578 125



```
                                CAR 1 CAR 2 CAR 3 CAR 4 CAR 5
no 0: [1, 2, 3, 4]     ——> 1111 0000 0000 0000 0000
no 1: [1, 2, 3, 8]     ——> 1110 0001 0000 0000 0000
no 2: [1, 12, 2,3]     ——> 1110 0000 0001 0000 0000
no 3: [1, 16, 2,3]     ——> 1110 0000 0000 0001 0000
no 4: [1, 2, 20,3]     ——> 1110 0000 0000 0000 0001
no 5: [1, 2, 4, 7]     ——> 1101 0010 0000 0000 0000
no 6: [1, 11, 2,4]     ——> 1101 0000 0010 0000 0000
...
no 624:[17,18,19,20]——> 0000 0000 0000 0000 1111
```

Figure 8.9: False VRP Combinations with Binary Representation (4 Landings to Visit).

Listing 8.9: True VRP Combination Generator.

```java
public int[][] get_real_order_combinations(Building ↩
    building, int[] valid_floors){
int[] valid_landings;
int floor_count;
int[][] false_VRP_combinations;
   valid_landings = valid_floors;
   floor_count = valid_landings.length;
   false_VRP_combinations = this.valid_combinations;
   final_VRP_cars = new int[false_VRP_combinations.length][↩
      valid_landings.length];
   real_VRP_combinations = new int[false_VRP_combinations.↩
      length][valid_landings.length];
   for (int i = 0; i < false_VRP_combinations.length; i++){
    for (int p = 0; p < floor_count; p++){
```

```
    final_VRP_cars[i][p] = ((false_VRP_combinations[i][p↩
        ]-1) / floor_count
      for(int c = 0; c < floor_count; c++) {
        for(int f = 0; f < floor_count; f++) {
          for(int mult = 0; mult < CAR_COUNT; mult++) {
            if (false_VRP_combinations[i][p] == (c+1 + ↩
                mult*floor_count) ) {
              real_VRP_combinations[i][p] = valid_landings[↩
                  c];
    }}}}}}
return real_VRP_combinations;
}
```

The brute force VRP algorithm presented in this section, requires tremendous amount of computational capacity. For every valid VRP combination a TSP algorithm extension is required per elevator car. This potentially translates to a minimum of 15000 additional iterations for a 4 floor BF VRP. The simulated combination that provides the best solution based on a cost function is kept and permanently allocated to each elevator car, where the rest of the combinations are discarded. Even if a tread is created for every VRP combination and its TSP extensions, it will proof to be unsuccessful if the number of requests increase. A possible solution to reduce the number of calculations without losing performance accuracy can be obtained by implementing a Genetic Algorithm in much the same way as with individual traveling salesman problem optimisations in Section 8.2.2.2.

8.3.3 UP-PEAK ELEVATOR CAR DISPATCHING WITHOUT DD

The up-peak traffic distribution is the most demanding period in a 24 hour elevator cycle and is often used as the benchmark for comparing elevator dispatching algorithms. With reference to Figure: 8.10, Table: 8.7 and Table: 8.8, the building population capacity is set to 500 people and the elevator configuration is defined as earlier, see Figure: 8.8. The VRP algorithm from the previous section is now tested and compared against various elevator car dispatching algorithms. Dispatching algorithm P1-A uses the principle of rotating allocated elevator cars between processed passenger requests. The idea is to keep all the elevator cars moving continuously throughout the building, if the requests are uniformly distributed then the cars will follow suit and vice versa. P1-B refer to a dispatching algorithm that will provide the next available elevator in the group. If none of elevator cars are in standby-state then the elevator car which is least used, based on number of services, is allocated to the new request. P1-C is similar to P1-B where the available elevator car is dispatched, but if none of elevator cars are in standby-state then the dispatching algorithm prioritise certain elevator cars. This will result in longer planned routes for certain elevator cars and fewer total services for other. The baseline algorithms: P1-A-Baseline, P1-B-Baseline, P1-C-Baseline, process requests in the same way as their namesake, but simulates the old group supervisory system based on the same directional logic as defined for a single elevator baseline simulation, see Section: 8.2.1. With Baseline directional logic, the landing floors are not prioritised based on a cost function but rather dependent on the direction the elevator car is traveling.

8.4 ELEVATOR GROUP CONTROL SIMULATION - WITH DESTINATION DISPATCHING

8.4.1 DESTINATION DISPATCHING VARIATIONS

The main philosophy for the destination dispatching (DD) system is when each passenger indicates his/her destination floor at arrival. The controller will then designate a specific elevator to that passenger, which will not change throughout the duration of the passenger's trip. This variation is for example implemented by Schindler's PORT system. The system does very well in high up-peak traffic periods proven by Figure 8.11 and 8.12, but is not ideal for other traffic periods. This leads to further variations to be investigated and compared against the main DD system. The BF-VRP can be extended to be used with DD control mechanisms as well. It outperformed the best non-computational DD and is a valid alternative to consider in practical application with the aid of GA to reduce computational requirements.

A variation to the main DD system is possible; where the designated elevator can alternate if a more optimal elevator has been identified by the controller. This means that while the passenger is waiting at the allocated car changes. Another variation is similar to the previous one, where the designated car can change, but in this variation; the passenger is required to physically move from one elevator to another at some point in his/her journey if so required. For example midway to the top floor of a building the passenger can be requested to swap elevators on route to the destination floor. This variation shall occur if it is optimally required by the controller or when the physical elevator shaft for that elevator does not reach the destination floor in the case of high rise buildings or underground facilities. The practical constraint is when the designated elevator car is changed; the information must be effectively communicated to the passenger. The communication can be done by giving each passenger a temporary number when he/she arrives at the landing and then the required information is displayed on a notice board. If an Integrated access control (IAC) system is operational the passenger's information can be used on the display board instead of a temporary number. If the passenger has access to any mobile device or Bluetooth enabled device like the latest digital watch technology, the information can be readily available and updated to each passenger's device. Another option and perhaps the best is that the elevator car number can be adjusted as required through a LCD screen above each elevator. For instance a passengers is directed to use elevator A to go to floor 65, but the physical location of designated elevator A can change in any moment with its LCD display as required. With additional limitations like not allowing the elevator number to change within 5 minutes of the elevator car arrival, can reduce passenger confusion and abrupt changes by the controller. With the third variation the passengers can be directed to travel with whichever elevator is classified as A all the way to the destination, even if he/she must move between elevators throughout the journey.

8.4.2 VRP DESTINATION DISPATCHING IMPLEMENTATION

In the case of multiple elevator cars being available for dispatching, we know that the VRP techniques will attempt to cluster passengers at the same floors together as well as floors which are in close proximity to each other to minimise cost. Another possibility is to create a pre-established movement distribution framework for one or more elevator cars. By doing this we can control the designated elevator car movements and ensure that certain floors are being prioritised individually by certain elevator cars and not necessary by the controller that is programmed by the various VRP techniques. Unfortunately by prioritising certain floors or passengers; will not have a positive impact on traveling cost or overall waiting time optimisation. But it will produce a more intelligent elevator controller that can emphasise customer satisfaction of certain passengers above overall efficiency.

A case study for a building population capacity of 500 was simulated with a few group control algorithms. Names for different control algorithms are defined as follows:

- P1: Refer to without destination dispatching (DD).

- BF VRP: Refer to a Brute Force Vehicle Rooting Problem algorithm or A, B, C that are other non-computational algorithm generally in use.

- GA2: Refer to Genetic Algorithm execution for the Traveling Salesman Problem execution per car or baseline with the old directional philosophies.

P1-BF-VRP-GA2 algorithm refers to the blue coloured plot in Figure 8.10 and executes the VRP computations for the simulated traffic. The P1-VRP algorithm performs well in comparison to the other elevator dispatching algorithms without DD in the up-peak state. The improvement percentage, with reference to the total waiting time, ranges from 7.7 % to 10.9 %. The Baseline algorithms expectantly lacked performance, with the P1-C elevator car dispatching algorithm performing the best from the tested non-computational algorithm generally in use today.

However the VRP algorithm with DD was the most optimal mechanism to implement in terms of service time. The destination dispatching algorithm improves total waiting time of the BF-VRP without DD by 21.3 % and 27.4 % from the baseline. The baseline landing waiting time was improved by 4 % and the time spend in the car was improved by 36.9 %, see Table 8.7 and 8.8. With DD-BF-VRP-GA2 the distance required to service the same amount of people is more, because there are less people in the elevator car at a time. Servicing less people at a time has resulted in 86 additional services from P1-VRP-BF. This means that DD is not more energy efficient than dispatching algorithms without DD.

The building population is now increased to a 1000 people with the same elevator configuration as before. The normal dispatching algorithms were not able to handle the high population demand versus the destination dispatching which was still capable of reaching an acceptable performance, see Figure 8.11 and Figure 8.12. Where DD improved total waiting time by 70 % at the cost of 86 additional services. When traffic demand reaches a certain level, energy considerations becomes irrelevant and passenger throughput maximisation becomes the dominating factor in elevator performance.

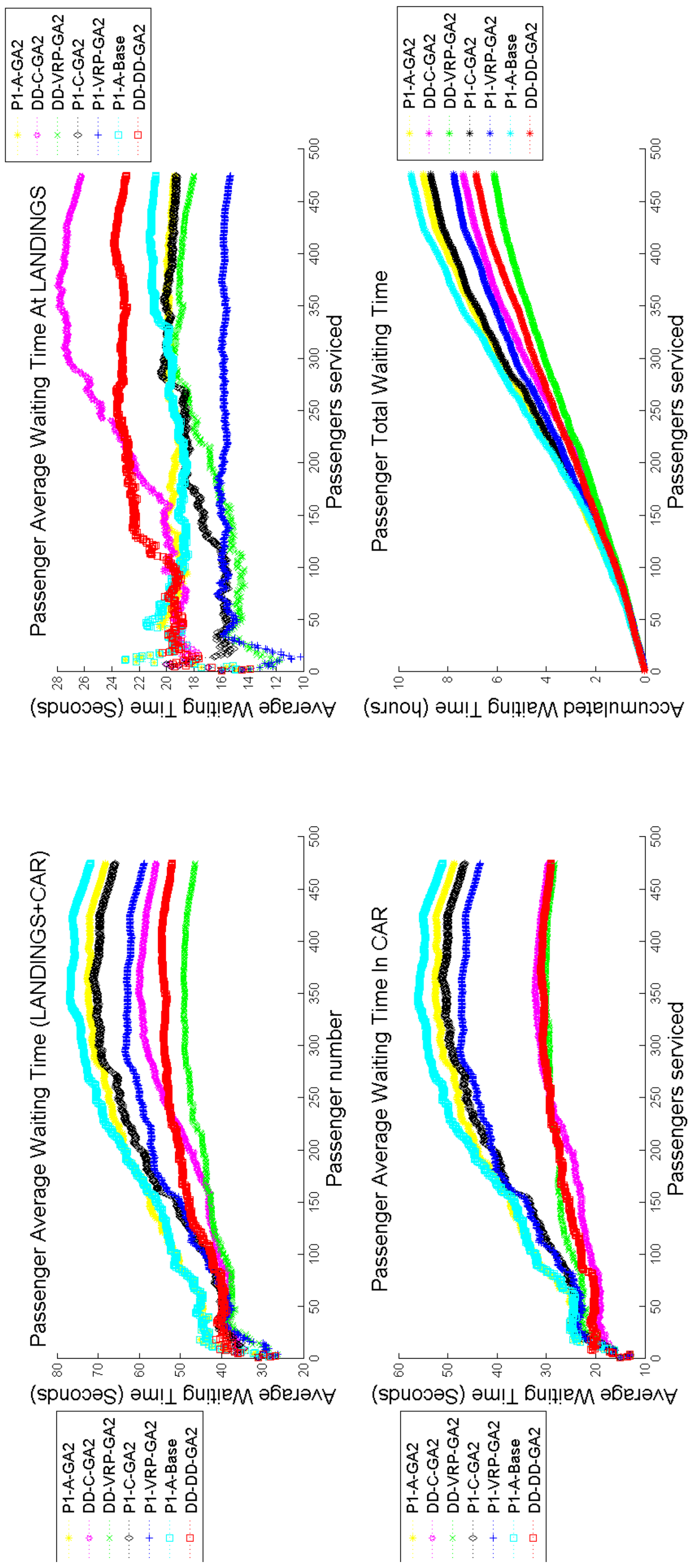


Figure 8.10: Service Waiting Time Performance for an Up-peak Cycle (Building Population Capacity of a 500 People).

Table 8.7: Elevator Controller Service Performance Results for a Group Elevator Configuration without Destination Dispatching and for a Building Population of 500.

	WD	WD	WD	**WD**	WD	WD	WD	WD
PHILOSOPHY	P1_A_GA2	P1_B_GA2	P1_C_GA2	**P1_VRP_BF**	P1_DD_GA2	P1_A_Base	P1_B_Base	P1_C_Base
TSP	GA2	GA2	GA2	**BF2**	GA2	Baseline	Baseline	Baseline
DISTANCE:								
CAR 0	1904 m	2096 m	4328 m	**2772 m**	1232 m	1836 m	1936 m	4728 m
CAR 1	1784 m	2156 m	1844 m	**2304 m**	376 m	2100 m	2204 m	2748 m
CAR 2	2072 m	1920 m	1652 m	**1832 m**	1120 m	1968 m	2276 m	1540 m
CAR 3	2316 m	2008 m	1528 m	**1524 m**	4160 m	1792 m	2072 m	928 m
CAR 4	2108 m	1764 m	1224 m	**1664 m**	904 m	2000 m	2016 m	820 m
TOTAL	10184 m	9944 m	10576 m	**10096 m**	7792 m	9696 m	10504 m	10764 m
STOPS:								
CAR 0	105	111	205	**151**	61	102	114	227
CAR 1	98	110	105	**127**	22	115	114	150
CAR 2	115	109	96	**119**	75	119	114	91
CAR 3	115	109	94	**95**	232	106	114	59
CAR 4	116	111	67	**79**	60	110	113	51
TOTAL	549	550	567	**571**	450	552	569	578
WT CAR	6h 25m 39s	6h 24m 44s	6h 9m 14s	**5h 43m 51s**	10h 32m 19s	6h 44m 38s	5h 57m 41s	5h 55m 56s
WT FLOOR	2h 34m 7s	2h 28m 14s	2h 32m 48s	**2h 1m 35s**	10h 44m 8s	2h 44m 41s	2h 33m 10s	2h 28m 35s
TOTAL	8h 59m 46s	8h 52m 58s	8h 42m 2s	**7h 45m 26s**	21h 16m 27s	9h 29m 19s	8h 30m 51s	8h 24m 31s
AVG WT CAR	0m 48s	0m 48s	0m 46s	**0m 43s**	1m 19s	0m 51s	0m 45s	0m 44s
AVG WT FLOOR	0m 19s	0m 18s	0m 19s	**0m 15s**	1m 21s	0m 20s	0m 19s	0m 18s
AVG TOTAL	1m 8s	1m 7s	1m 5s	**0m 58s**	2m 40s	1m 11s	1m 4s	1m 3s
VRP	0	0	0	**3595**		0	0	0
TSP	8110	7761	7498	**20271**	7010	0	0	0

Table 8.8: Elevator Controller Service Performance Results for a Group Elevator Configuration with Destination Dispatching and for a Building Population of 500.

	DD	DD	DD	DD	DD	**DD**
PHILOSOPHY	DD_A_GA2	DD_B_GA2	DD_C_GA2	DD_DO_GA2	DD_DD_GA2	**DD_VRP_GA2**
TSP	GA2	GA2	GA2	GA2	GA2	**GA2**
DISTANCE:						
CAR 0	3828 m	3312 m	5456 m	5544 m	1472 m	**2768 m**
CAR 1	3392 m	3676 m	3876 m	464 m	2108 m	**3260 m**
CAR 2	3576 m	3392 m	3236 m	444 m	2784 m	**3244 m**
CAR 3	3472 m	3348 m	2612 m	520 m	3904 m	**2544 m**
CAR 4	3492 m	2944 m	2420 m	844 m	4360 m	**2208 m**
TOTAL	17760 m	16672 m	17600 m	7816 m	14628 m	**14024 m**
STOPS:						
CAR 0	145	137	234	323	133	**154**
CAR 1	150	138	157	26	141	**158**
CAR 2	150	137	127	32	126	**154**
CAR 3	142	137	115	22	125	**108**
CAR 4	147	139	98	34	113	**83**
TOTAL	734	688	731	437	638	**657**
WT CAR	3h 53m 55s	6h 20m 33s	3h 54m 28s	12h 34m 12s	3h 49m 49s	**3h 44m 18s**
WT FLOOR	3h 35m 54s	10h 36m 1s	3h 28m 3s	6D 6h 43m 33s	3h 1m 50s	**2h 22m 28s**
TOTAL	7h 29m 49s	16h 56m 34s	7h 22m 31s	6D 19h 17m 45s	6h 51m 39s	**6h 6m 46s**
AVG WT CAR	0m 29s	0m 47s	0m 29s	1m 35s	0m 28s	**0m 28s**
AVG WT FLOOR	0m 27s	1m 20s	0m 26s	18m 59s	0m 22s	**0m 17s**
AVG TOTAL	0m 56s	2m 8s	0m 55s	20m 35s	0m 51s	**0m 46s**
VRP	0	0	0	0	0	**2380**
TSP	13983	13383	13917	8691	11795	**18124**

8.5 SUMMARY AND CONCLUSIONS

The various machine learning techniques and AI concepts discussed throughout this Thesis was implemented in this chapter. However to develop, test and compare any new elevator control philosophies a fully functional testing platform was created. The virtual environment for the elevator application includes the relevant building dynamics together with simulated population distributions. Three different traffic generators were implemented, namely a Complete Randomisation method and two Expert System biased population generators.

In the chapter three different elevator control configurations were simulated, namely single elevator car control, elevator group control without destination dispatching (DD) and elevator group control with DD. Each configuration included different control algorithms that were compared with each other and with the newly developed control mechanisms.

In the single elevator car control simulation, we established a baseline from an actual installation and proposed a few improvements to the control philosophy by means of the Traveling Salesman Problem (TSP) solutions. The TSP was implemented and tested with a couple of solution attempts, namely with Brute Force (BF), Genetic Algorithms (GA) and by Simulated Annealing. According to Krumke [16], the best possible online algorithm has a competitive ratio of 2 and the BF and the GA algorithms implemented in this section compares well with the stated benchmark for a single elevator application, with 2.17 and 2.18 respectively. The baseline control being implemented by the present configuration, were proven to be the least competitive with a competitive ratio of 3.55 and an approximation ratio of double that of the best possible solution. The approach to solve the TSP with Simulated Annealing was proven to be unsuccessful, because simulated annealing can only be implemented if the cost calculations are between non-spatial variables. For a specific test case we were able to reduce the number of calculation from 24 million by the BF TSP to about 7 million for the GA TSP. Thus as a result of machine learning techniques like Genetic Algorithms; the overall number of iterations and computational time have been reduced considerably and have not effected the performance negatively with comparison to the Brute Force approach.

An actual case study was conducted for Majuba Power Station which has 6 similar elevators spread out over the 6 generating units. Table 8.5 provides the developed Energy Model results for a 5 day period. The TSP solutions improved the waiting times at the landings by more than 24% and the total waiting time by 20% over the actual installation results. However the power consumption over the 5 days were not reduced by the TSP solutions, because waiting time had priority over energy savings in this instance. The results

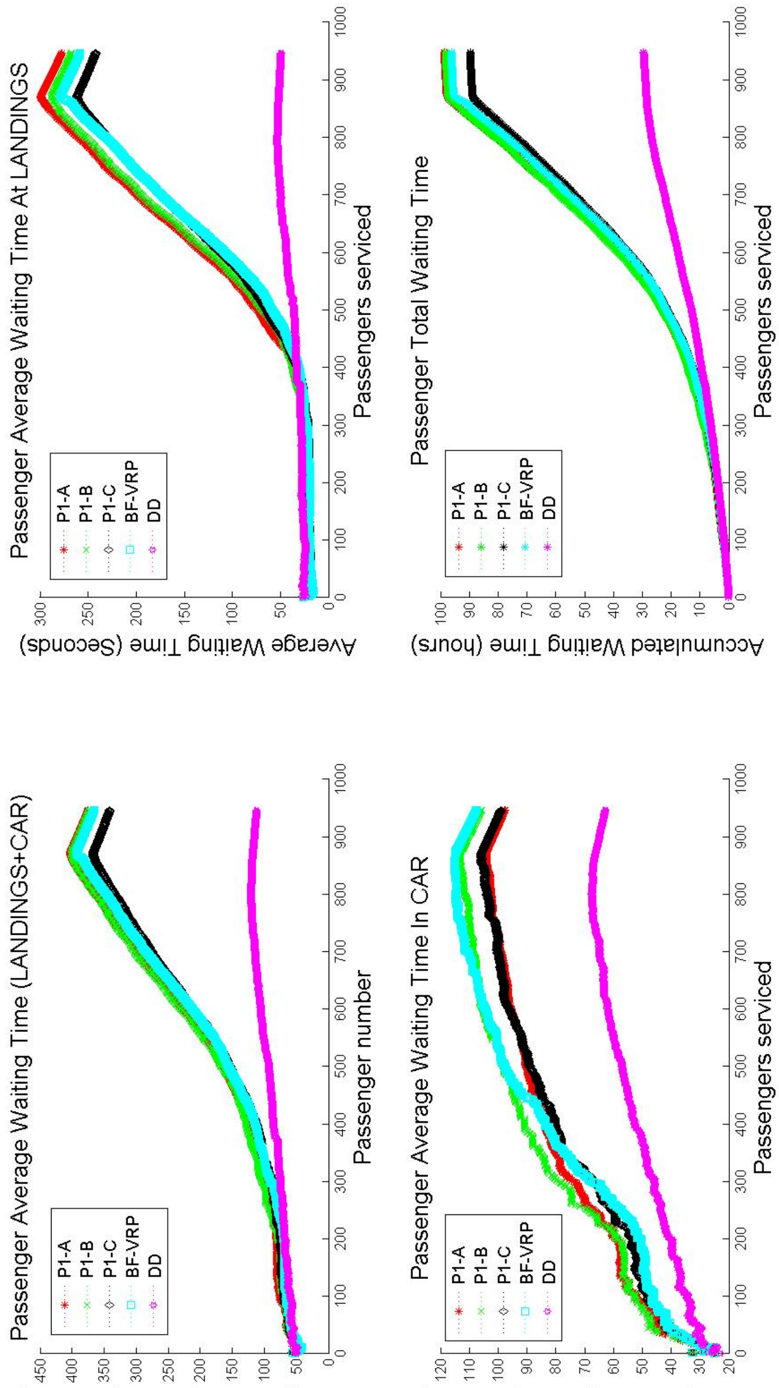


Figure 8.11: Service Waiting Time Performance for an Up-peak Cycle (Building Population Capacity of a 1000 People).

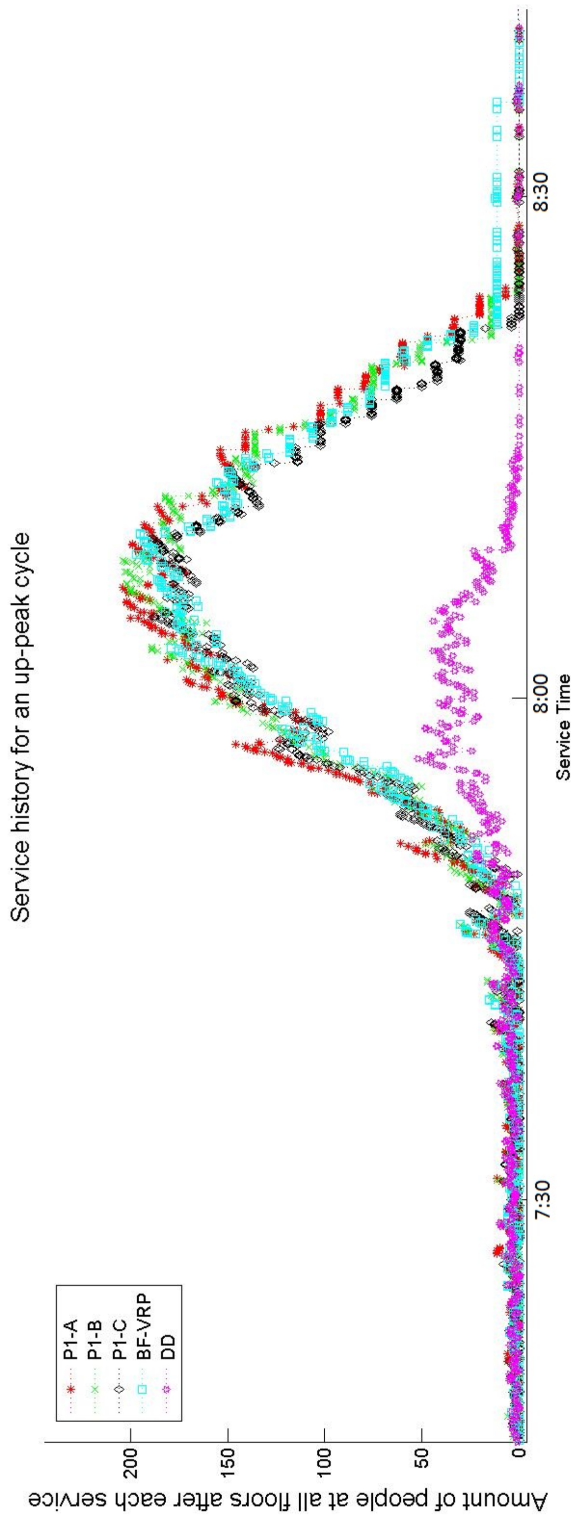


Figure 8.12: Service Passenger Count Performance for an Up-peak Cycle (Building Population Capacity of a 1000 People).

from the case study, also provided usage indicators, for example the elevator car was stationary for 86.3% of the time and the total accumulated distance reached 46 km with a total power consumption of 277 kWh. With 6 similar elevators the projected expenditure for a year is around R120 000, with 80 MWh, 1.3 million stops and 13 000 km expected distance travelled. It was also concluded that by upgrading the actual elevator installation to be able to handle regenerative braking, the potential energy savings can be more than 28% on average excluding the additional energy that can be reverted back into the system as well. The actual minimum savings per trip was measured at 10.86% from 20 to 0m and at a maximum of 65.12% from 16m to 20m, which measured at 31.79kJ and 140.74kJ respectively.

A simulated case study for a building population capacity of 500 was simulated with a few elevator group control algorithms. During the up-peak state the VRP without DD improved the baseline waiting time by 8 %. However the VRP algorithm with DD was the most optimal mechanism to implement in terms of service time but not in terms of energy efficiency. The destination dispatching algorithm improves total waiting time of the BF-VRP without DD by 21.3 % and 27.4 % from the baseline. With DD-BF-VRP-GA2 the distance required to service the same amount of people is more, because there are less people in the elevator car at a time. Servicing less people at a time has resulted in 86 additional services from P1-VRP-BF. This proves that DD is not more energy efficient than dispatching algorithms without DD. Destination dispatching technology does very well in high up-peak traffic periods proven by Figure 8.11 and 8.12, but is not ideal for other traffic periods. This leads to further variations to be investigated and compared against the main DD system. The BF-VRP was extended to be used with DD control mechanisms as well. It outperformed the best non-computational DD in all the other traffic periods and is a valid alternative to consider in practical applications with the aid of GA to reduce computational requirements.

Chapter 9

CONCLUSIONS AND RECOMMENDATIONS

9.1 SUMMARY

The work presented in this thesis focused on the potential of machine learning solutions in relation to the elevator control application, where traffic flow is controlled and directed according to certain control philosophies. The primary objective for this thesis was to investigate the potential of a machine learning solution and the relevancy of such technology in an elevator control application. The aim was to establish how the research field of machine learning and neural network science can be successfully utilised with the goal of creating an artificial intelligent (AI) controller. The AI controller is to adapt to its existing state and change its control parameters as required without the intervention of the user.

The secondary objective for this thesis was to develop an elevator model that represents every aspect of the real-world application. The model was to improve the accuracy of existing theoretical and simulated models, by modulating previously unknown and complex variables and constraints. The aim was to create a complete and fully functional testing platform for developing new elevator control philosophies and testing new elevator control mechanisms.

To achieve these objectives, the main focus was directed to how waiting time, probability theory and power consumption can be optimally utilised by means of machine learning solutions. The theoretical background was provided for these concepts and how each subject can potentially influence the decision making process. The reason why this approach has been difficult to implement in the past, is possibly mainly due to the lack of adequate representation for these concepts in an online environment without the continuous feedback from an Expert System. As a result of this thesis, the respective online models for each of these concepts were successfully developed in order to deal with the identified shortcomings.

The developed online models for projected waiting times, probability networks and power consumption feedback were then combined to form a new Intelligent Elevator Control (IEC) structure as opposed to the Expert System approach mostly used in present computer based elevator controllers.

9.2 THESIS CONTRIBUTION AND ORIGINAL CONTENT

This thesis reiterated the importance of machine learning techniques in the field of elevator control. An original software structure for computer based elevator group control was introduced to incorporate the developed online models for projected waiting time, probability networks and power consumption feedback. Combined with the computational routing mechanisms, in the form of the Traveling Salesmen Problem (TSP) and the Vehicle Routing Problem (VRP) solutions.

This thesis produced a more accurate and reliable approach to the cost function calculations, mostly used for the TSP and VRP optimisation algorithms. A Neural Elevator Model was developed to provide accurate power consumption feedback and running time predictions based on the exact elevator configuration responses.

Lastly this thesis developed an original testing platform and an elevator simulator, where existing and new control philosophies can be tested and compared offline. The possibility of an online testing platform was also investigated, where a more adaptable controller is created, which can learn any new algorithms on the go. The controller has the potential of automatically reconsolidating when or if such an algorithm should be implemented and if it will improve its current performance. To implement the online testing platform, the concept of a Neural Automated Reasoning Head (NARH) was developed. Any new control algorithm can be fed into the NARH together with its instruction set, respond cards and predefined conditions. In practice the online NARH can potentially recognise an input traffic demand pattern and then selects the best online control algorithm that it has been trained with. The chosen algorithm is then applied to the unprocessed requests together with the requests still in the system to produce the most optimal route.

9.3 CONCLUSIONS

A brief introduction to elevator control possibilities and background information was provided in this thesis. Mention is made to online vs. offline strategies and the implementation thereof. Various machine learning concepts and artificial intelligence philosophies were introduced with the intent of implementing some of them in an elevator application. Specific references were made to Neural Networks, Genetic Algorithms and Simulating Annealing.

The general power consumption philosophy is to create an energy model for our elevator configuration based on theoretical calculations and actual energy samples. The Measurement and Verification (M&V) concept was introduced where different procedures to obtain the actual energy samples were conducted and compared to each other. The best procedure was proven to be the method where the measurements are directly taken from the machine room equipment, where voltage drop can be neglected and additional measurement points are available. The actual energy samples were then compared to the theoretical calculations based on the rated nameplate values, as well as the developed equations. The overall system efficiency was concluded to be around 49% for a specific test case: 2 speed motor geared traction machine. The exact operations of the two speed motor were discussed and directly concluded from the graphical representations and compared to the equivalent single phase circuit equations. It was realised that each approach to create an accurate energy model had a major shortcoming that needed to be addressed. With any theoretical model, there is always uncertainties and unknown variables, which are most often neglected and ignored. Thus exact equations are very difficult to obtain and to solve. The shortcoming with energy sampling, is that it is not always feasible to obtain all permutations, for it can be very time consuming and sometimes difficult to conduct during busy working environments. For example for a 10 storey building, the number of required samples reaches 900 for all known sample permutations. A solution to the identified shortcomings can be found with various machine learning techniques, where the required samples can be severely reduced to only a few representative samples. The actual samples have been utilised in conjunction with a created Neural Network to establish true trending and accurate distribution patterns across the different combinations and load percentages. With the implementation of Neural Networks, the unknown variables can be accurately modelled without the need to define them. The created energy model was then ultimately used to estimate the energy consumption for any floor order route and for accurately predicting the actual running times of an elevator car, without making any assumptions, as it would have been the case with a theoretical model. The energy model was successfully incorporated in the TSP and the VRP, where the shortest route is calculated between the possible floor orders.

A probability model has been developed based on the probability philosophy definition and various theoretical predictions. The basic philosophy is to process previous collected traffic data and to use it to estimate future traffic occurrences to create pre-planned elevator car responses to the probability of future passenger requests. The probabilistic traveling salesman problem (PTSP) was defined, which is similar to the classical TSP, but with service probabilities, assuming independence between floors. The PTSP is attempting to deal with the uncertainty of routing problems, like nondeterministic cost calculations and uncertainty in passenger demand at each floor. References were made to well-known probability theorems like the Bernoulli trail probability and the Bayesian theorem that connects the respective prior floor probabilities with the posterior floor probabilities. The Poisson arrival probability function was implemented, with λ_i defined as the average number of passenger arrivals per floor in a specific period T. The general Poisson probability density $f(x)$ and the Poisson probability distribution $F(x)$ were used to provide a distribution model for stochastic passenger arrivals at each floor. Through the non-homogeneous Poisson process the directional and up-peak expected requests could also be established. A technique was developed to obtain the best suited Poisson rates for each traffic period with the introduction of a cost function. The expected distance of travel and the expected car capacity probabilities also contribute to the TSP algorithm together with the VRP, to generate a travel route which will reduce overall traveling costs and maximise throughput. A neural probability model was also developed later in the thesis, in conjunction with the relevant theoretical references.

The general waiting time philosophy was provided, which states that the controller should be able to minimise overall waiting time of all passengers by establishing the most optimal route to follow for each car. The concept of time management was also introduced, instead of just looking at time minimisation techniques. An equation was developed for any projected floor order list to provide the expected passenger waiting time and to be used in any TSP route costing calculations. A waiting time conjoining network was developed, by implementing radial base function (RBF) techniques. An RBF network is based on the approximation of an arbitrary continuous function from accumulated data points or from linear superposition of localised basis functions. The RBF network was used for exact interpolation between waiting times from various individuals and groups to illustrate the influences of accumulated waiting times across the building. The interpolated samples can be used by the controller to prioritise between which one of the floors to service next as a result of the accumulated influence from surrounding floors. It also resulted in the positioning of cars to be in close proximity to the next floor to service and the one after that. The RBF network can also be used for dynamic zoning when a clustering algorithm is implemented for destination dispatching purposes.

When it comes to an elevator application, data is not always accurate or complete and it creates a case of imprecision. The issue of uncertainty is addressed with probability and Expert System theory as discussed in this thesis, but also with the help of neural networks. Neural networks were implemented to deliver three types of solutions, which is classification, pattern recognition and prediction. The elevator controller application had a few classification requirements, for instance to establish the present state of the elevator and the available car capacity against pre-set arrangements. Other elements that required classification were the respective traffic patterns and passenger demand intensities during a time period. By using membership functions it was proven that we can classify the fuzziness or imprecise approximations of any discreet or continuous elements. Well trained neural networks were used for the membership mapping of the data points and was effectively checked against a test set. With changing traffic patterns it is important that the system continuously produce and implement better solutions to deal with the changing environment. Neural networks were implemented as a pattern recognition solution to develop an accurate energy model representation together with accurate running time modelling against different distances and load variations. Neural Networks were also utilised to model the passenger distribution across the building and to predict passenger demand throughout any scalable period of time. Lastly, automated reasoning principles were introduced and the possibility of an online testing platform were investigated, which can learn any new algorithms and automatically reconsolidate when or if such algorithm should be implemented to improve the ultimate performance of the controller.

The various machine learning techniques and AI concepts discussed throughout this thesis was implemented in the elevator control simulation chapter, where a fully functional testing platform was created. The virtual environment for the elevator application includes the relevant building dynamics together with simulated population distributions. Three different traffic generators were implemented, namely a Complete Randomisation method and two Expert System biased population generators. Three different elevator control configurations were simulated, namely the single elevator car control, elevator group control without destination dispatching (DD) and elevator group control with DD. Each configuration included different control algorithms that were compared with each other and with the newly developed control mechanisms. In the single elevator car control simulation, we established a baseline from an actual installation and proposed a few improvements to the control philosophy by means of the Traveling Salesman Problem (TSP) solutions. The TSP was implemented and tested with a couple of solution attempts, namely with Brute Force (BF), Genetic Algorithms (GA) and by Simulated Annealing. According to Krumke [16], the best possible online algorithm has a competitive ratio of 2 and the BF and the GA algorithms implemented compared well with the stated benchmark for a single elevator application, with 2.17 and 2.18 respectively. The baseline control of the actual case study configuration, were proven to be the least competitive with a competitive ratio of 3.55 and an approximation ratio of double that of the best possible solution. The approach to solve the TSP with Simulated Annealing was proven to be unsuccessful, because simulated annealing can only be implemented if the cost calculations are between non-spatial variables. For a specific test case we were able to reduce the number of calculations from 24 million by the BF TSP to about 7 million for the GA TSP. Thus as a result of machine learning techniques like Generic Algorithms, the overall number of iterations and computational time have been reduced considerably and have not effected the performance negatively with comparison to the Brute Force approach.

An actual case study was conducted for Majuba Power Station which has 6 similar elevators spread out over the 6 generating units. A simulation model was created for a 5 day period, where the TSP solutions improved the waiting times at the landings by more than 24% and the total waiting time by 20% over the actual installation results. However the power consumption over the 5 days were not reduced by the TSP solutions, because waiting time had priority over energy savings in this instance. The results from the case study, also provided usage indicators, for example the elevator car was stationary for 86.3% of the time and the total accumulated distance reached 46 km with a total power consumption of 277 kWh. With 6 similar elevators the projected expenditure for a year is around R120 000, with 80 MWh, 1.3 million stops and 13 000 km expected distance travelled. It was also concluded that by upgrading the actual elevator installation to be able to handle regenerative braking, the potential energy savings can be more than 28% on average excluding the additional energy that can be reverted back into the system as well. The actual minimum savings per trip was measured at 10.86% from 20 to 0m and at a maximum of 65.12% from 16m to 20m, which measured at 31.79kJ and 140.74kJ respectively.

A simulated case study for a building population capacity of 500 was simulated with a few elevator group control algorithms. During the up-peak state the VRP without DD improved the baseline waiting time by 8 %. However the VRP algorithm with DD was the most optimal mechanism to implement in terms of service time but not in terms of energy efficiency. The destination dispatching algorithm improves total waiting time of the BF-VRP without DD by 21.3 % and 27.4 % from the baseline. With DD-BF-VRP-GA2 the distance required to service the same amount of people is more, because there are less people in the elevator car at a time. Servicing less people at a time has resulted in 86 additional services from P1-VRP-BF. This proves that DD is not more energy efficient than dispatching algorithms without DD. Destination dispatching technology does very well in high up-peak traffic periods proven by Figure 8.11 and 8.12, but is not ideal for other traffic periods. This leads to further variations to be investigated and compared against the main DD system. The BF-VRP was extended to be used with DD control mechanisms as well. It outperformed the best non-computational DD in all the other traffic periods and is a valid alternative to consider in practical applications with the aid of GA to reduce computational requirements.

9.4 RECOMMENDATIONS FOR FUTURE WORK

The work presented in this thesis focused more on the potential of machine learning solutions in relation to the elevator control application, rather than the actual performance results. It is difficult to compare any new control algorithms with the baseline algorithms in practice today, because the exact control functions are not always public knowledge. However the proposed solutions did show great potential and should be investigated further. Alternative variations and new machine learning solutions can also be implemented in the elevator control application and require further investigation. The newly proposed Intelligent Elevator Controller (IEC) software structure is still in the infant stage and requires further development into a workable model for actual testing. The online testing platform concept shows great potential and should be further developed. Lastly the principles and techniques that were covered by this thesis can be extended to other control applications, especially for online routing applications. Applications with varying loads can also benefit from the same energy modelling techniques presented in this thesis, for example pump configurations and wind turbines.

List of References

[1] Barney, G.C.: Lift traffic analysis design and control. *Stevenage, Eng. : Peter Peregrinus Ltd.*

[2] Bishop, C.: Neural networks and their applications.

[3] Heaton, J.: Introduction to neural networks with java. p. 372, 2005.

[4] Kalra, P.: Advanced source/drain technologies for nanoscale cmo,.

[5] Potvin, J.-Y.: The traveling salesman problem: A neural network perspective.

[6] Groenewald, Nejatian, A and Wilson, J.: Specification for three phase energy metering for use with measurement and verification projects, measurement and verification.

[7] Godie, A.: C1 IEE installations project. 2014.

[8] Wildi, T.: Electrical machines drives, and power systems. vol. Sixth Edition.

[9] Umans, S.: Fitzgerald & kingsley's electric machinery. vol. Seveth Edition.

[10] Shimazak, H. and Shinomot, S.: A method for selecting the bin size of a time histogram. *Neural Computation 19*, p. 1503 to 1527, 2007.

[11] Barney, G.: Elevator traffic handbook. 2004.

[12] S., D. and Papadimitriou, C.: NP-complete problems.

[13] Asvestopoulos, L. and Spyropoulos, N.: Lifts energy consumption study. vol. Category: Issue 5/2010, 2010.

[14] Mizon, J.: Transit management: Now it's possible.

[15] Official schindler website: http://www.schindler.com.

[16] Krumke, S.O.: Online optimization competitive analysis and beyond. 2001.

[17] Soluade and Oredola, A.: Modelling and simulation of a resource allocation problem. vol. Communications of the IIMA , Vol. 12, No. 1.

[18] Potvin, J.-Y. and Thangiah, S.: Vehicle routing through simulation of natural processes.

[19] Hornik, K. and Hahsler, M.: TSP infrastructure for the traveling salesperson problem.

[20] Koehler, J. and Ottiger, D.: An ai-based approach to destination control in. *AI Magazine*, vol. Volume 23 Number 3, 2002.

[21] Akyol, Y., Buyukkaracigan, B. and Nuri, M.: The importance of fuzzy logic approach in lift system control of high-rise buildings. 2011.

[22] Vidhya, B., Rajasulochana, P., P.B.R., B. and Krishnamoorth, P.: evolutionary study of moray eel with in-silico drug design. *Research Journal of Pharmaceutical, Biological and Chemical sciences. ISSN: 0975-8585*, vol. Volume 4 Issue 3 Page No. 176, p. 7, 2013.

[23] Lewontin, R.: The units of selection annual review of ecology and systematics. vol. Vol. 1: 1-18 (Volume publication date November 1970).

[24] Lotz, M.: Modelling of process system with genetic programming. 2006.

[25] Paterlini, S. and Minerva, T.: Regression model selection using genetic algorithm, recent advances in neural networks, fuzzy systems and evolutionary computing.

[26] Silva, S.: A genetic programming toolbox for Matlab. 2004.

[27] Shi, Z.: Advanced artificial intelligence. *Singapore ; World Scientific, Hackensack, NJ*, 2011.

[28] Golden, B., Raghavan, S. and Wasil, E.: The vehicle routing problem : latest advances and new challenges. 2008.

[29] Chapman, S.: Electric machinery fundamentals. vol. 4th edition, McRaw Hill international edition, 2005.

[30] Al-Sharif, L., Peters, R. and Smith, R.: Elevator energy simulation model.

[31] White, R.: Electromechanical elevator. p. 43, 2007.

[32] Lutfi, R.D.: Mechanical basics, selected topics of mechatronics.

[33] OMICRON CMC 256 plus, user manual.

[34] Bhatia, A.: Understanding motor nameplate information: Nema vs. iec standards. p. 29.

[35] Sachs, H.: Opportunities for elevator energy efficiency improvements. *American Council for an Energy-Efficient Economy,*, 2005.

[36] Jaillet, P.: A priori solution of a traveling salesman problem.

[37] Peebles, P.Z.J.: Probability, random variables and random signal principles. vol. 4th edition, p. 462, 2001.

[38] Gallager, R.G.: Discrete stochastic processes. vol. draft of 2nd Edition, 2011.

[39] Timothy, J.: Fuzzy logic with engineering applications. 2010.

[40] Lathi, B.P.: Modern digital and analog communication systems. vol. third edition, p. 781, 1998.

[41] Carandini, M. and Heeger, D.J.: Normalization as a canonical neural computation. *Nature Reviews, Neuroscience*, vol. 13, 2012.

[42] Etzkorn, B.: Data normalization and standardization. 2012.

[43] LeCun, Y., Bottou, L., Orr, G. and Muller, K.-R.: Efficient backprop. *Springer*, 1998.

[44] Ming Leung, K.: Preparing the data. 2007.