FUZZY LOGIC LOAD FORECASTING WITH GENETIC ALGORITHM PARAMETER ADJUSTMENT

Craig Stuart Carlson

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 2012

Declaration

I declare that this dissertation is my own, unaided work, other than where specifically acknowledged. It is being submitted for the degree of Master of Science in Engineering to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other university.

Signed this _____ day of _____ 2012

_____

Craig Stuart Carlson

Abstract

World-wide pressure on existing power distribution systems calls for action to be taken in order to curb the energy deficit. The concept of a smart grid can assist since a significant function is the improvement of energy efficiency in transmission and usage. This is also known as energy management. Load forecasting can indirectly aid energy management by raising user awareness to reduce the peak and total power usage. Load forecasting has been implemented using many different methods over the years, from statistical methods to computational intelligence methods. Combinations of methods also exist to enhance the forecasting capabilities. Following from observations made, it was hypothesised that a fuzzy logic load forecasting algorithm could be improved by incorporating an optimisation technique such as genetic algorithms.

In order to observe the effects of a genetic algorithm on a fuzzy logic load forecasting system, MATLAB® was used to implement a load forecasting algorithm using fuzzy logic systems and genetic algorithms. The fuzzy logic systems used the day (week or weekend), the time of day and the historic power usage to perform the forecasting. The genetic algorithm adjusted the fuzzy logic parameters to minimise the peak and total energy errors in a 24 hour period.

Using data from one week prior to the test yielded the most accurate results after considering varying quantities of input data. The results obtained from five case studies indicated a good correlation between the forecast and measured values. Initial results were obtained using *a priori* knowledge of the behaviour of the system, then the genetic algorithm was implemented. The full week forecast results showed an average improvement, for the five cases, of 4.32 and 18.95 times for the peak energy error and the total energy error respectively. This indicates that the dissertation hypothesis was proven to be correct.

To my ever-supportive parents and my loving wife

for your endless support and for constantly pushing me to fulfil my dreams

Acknowledgements

This research was performed under the auspices of the Future Electrical Engineering Technologies research group as well as the Control Research Group at the University of the Witwatersrand, Johannesburg South Africa.

Firstly, and most importantly, thanks are extended to the South African power producer, Eskom, for without their support and funding this research would not have been possible. Additional thanks are extended for their generous supply of the load profile data from the Southern Region (Eastern Cape Province) used in this research.

Next to my supervisors, Prof. W.A. Cronjé and Prof. M.A. van Wyk, thank you for letting me discover my own way through this big world that is postgraduate life. You let me gain the experience, discipline and knowledge required to complete something of this level in my own time but pushed me in the right direction when I was going astray. Without your gentle prodding I fear I would still be at the beginning!

Of course, how could I forget my colleagues, friends and family? You have stood by me through all the highs and lows. You gave me more minds to bounce ideas off. For this I cannot express enough gratitude.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The existing power distribution systems world-wide are currently experiencing energy deficits. This necessitates a rapid enhancement of the existing system architecture. One way in which this can be achieved is by implementing the concept of a smart grid. Functional requirements of a smart grid include the more effective distribution and use of available power, which can be considered energy management. This can be achieved by implementing a means of control over the total energy and the peak energy usage. Load forecasting could be used indirectly to create user awareness to the total and peak power usage and thus aid in energy management. Load forecasting has been implemented for many years using different techniques ranging from statistical methods to various computational methods.

This dissertation hypothesises that a fuzzy logic system used for load forecasting will be improved (implying a reduction in errors) by incorporating an optimisation technique such as genetic algorithms.

This dissertation is structured as follows:
Chapter 2 introduces the background pertaining to load forecasting and its place in research today. The concept of a smart grid is discussed as well as the subsequent result of energy management. It is hypothesised that load forecasting can help achieve the required energy management and thus several different methods of load forecast are discussed.

Chapter 3 provides a detailed description of the development and implementation of the load forecasting algorithm and associated features using fuzzy logic systems

and genetic algorithms. The necessary assumptions and constraints to enable implementation are presented and the performance criteria are described.

Chapter 4 begins with the tests and results determining the required amount of input data to yield accurate results. Detailed case studies are then presented to show the effect that the genetic algorithm has on the load forecasting results. Analysis of the results finalise the research findings.

Chapter 5 proposes several factors of the algorithm structure to alter to be investigated further. The purpose of the additional investigation would be to observe the effects on the performance of the load forecasting algorithm. Several recommendations are also proposed in order to potentially improve the existing performance criteria and provide additional knowledge about the algorithm performance. Simple tests are performed were possible to substantiate the recommendations.

Chapter 6 concludes the dissertation and provides a summation of the most pertinent results obtained in the research.

Appendix A details and compares previously implemented load forecasting methods across a range of categories such as the requirements; the forecast period and the mean absolute percentage error. This is in support of Chapter 2.

Appendix B provides important definitions pertaining to fuzzy logic and genetic algorithms, as required for Chapter 3.

Appendix C documents the MATLAB® scripts that were developed for the load forecasting algorithm. This is in support of Chapter 4.

Appendix D contains additional maps, graphs and other data to enhance the understanding of the case studies performed in Chapter 4.

Chapter 2

Background

Chapter Overview: *The first step to understanding any problem is to establish and comprehend existing information pertaining to the problem. This chapter presents information leading to the development of the research topic in load forecasting. A brief description of the current grid architecture is covered, followed by an overview of a smart grid concept as well as descriptions of several of the existing methods for load forecasting.*

2.1 Current Grid Architecture

Current power distribution systems are built based on an architecture that has been used for over a century [1]. A depiction the current power distribution system is shown in Figure 2.1 [2]. Quality of service and reliability is maintained by creating sufficient generation and distribution with surplus capacity to allow for fluctuations and growth [1].

This architecture needs to be enhanced since there is an increase in power demand world-wide [3] and is witnessed in South Africa by the increasing number of regional black-outs [4]. These black-outs are due to surplus power reserves being reduced to below 10 % while a healthy reserve is stated to be between 17 - 20 % [5]. Another indication that an enhancement of the current power distribution system is necessary can be seen in the increasing pollution levels and other environmental impacts [1, 3] that need to be counteracted.

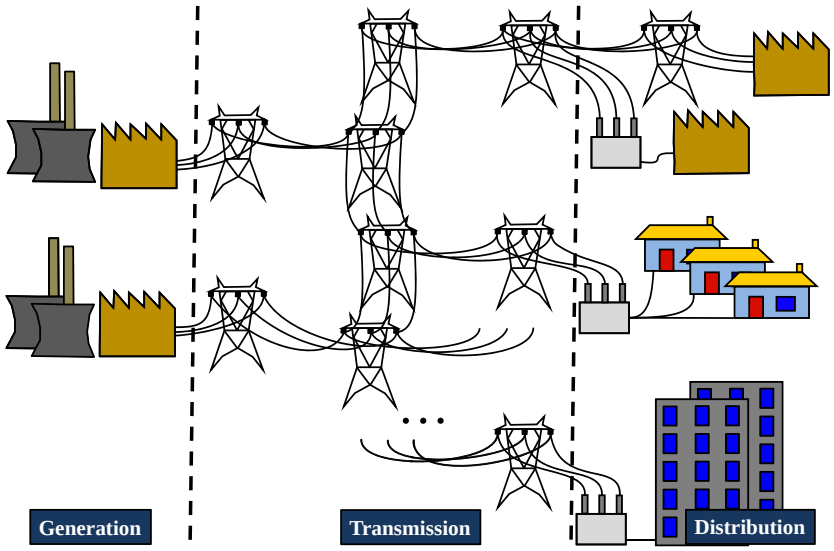


Figure 2.1: A representation of the existing power distribution system architecture.

2.2 Overview of a Smart Grid

Although there are many different interpretations of the smart grid concept, there is no single unified solution in order to implement a smart grid. Based on current literature the power distribution system of the future, or what is considered a smart grid, enhances the current power transmission grid by incorporating monitoring, control and communications [2, 6, 7].

By incorporating the additional functionality, the consensus as to the functions a smart grid should be capable of includes [1, 2, 6]:

1. More efficient and optimised use of available assets,
2. The introduction of customer participation,
3. The ability to incorporate all distributed generation and energy storage systems,
4. Improved quality of service,
5. Self-healing by rapid response to disturbances in the system,
6. Resistance against man-made attacks and natural disasters and
7. The allowance for new products, services and markets to emerge.

The first function can be achieved by introducing energy management, and incorporating the second function. Energy management is an important factor in the world today since it focuses on increasing the efficiency of most energy consuming processes [8]. Thus, energy management can be used to monitor and control the efficiency of the usage and distribution of power. Load forecasting is not the solution to the energy management issue, but a tool in its implementation [9]. Using the requirements of energy management as a base, load forecasting can be used to indirectly improve the usage of power by raising user awareness and making the end-process more efficient (assuming the user provides the necessary intervention) [10]. The distribution of energy can be made more efficient, again indirectly, due to load forecasting. The power producer can forecast what the load would be and ensure that a sufficient quantity is provided.

2.3 Overview of Existing Methods of Load Forecasting

Load forecasting is the prediction of the power usage for a chosen area, which can be implemented using several different techniques that include:

- Statistical models (Such as regression models and time series models).
- Computational intelligence models (Such as expert systems; artificial neural networks; fuzzy logic systems).
- Combinations of methods

2.3.1 Statistical Models

Statistical models for load forecasting can be broadly classified into two categories depending on how the load is treated. The first method assumes the pattern of the load is a time series signal and can thus perform time series analysis to obtain the desired result. The second method strives to obtain the relationship between power usage and weather variables, using regression analysis [11, 12].

Time Series Models

The power usage could be approximated as a time series since it can be assumed to be periodic on various scales (such as daily, weekly and seasonally). This approximation is accurate so long as the periodicity holds true. When fluctuation is introduced then the time series analysis fails. The Box-Jenkins method is considered to be the most efficient forecasting technique assuming the analysed time series is stationary [13, 14].

Regression Analysis Models

Regression analysis effectively performs curve fitting to manipulate the output to the desired value [14]. In order to implement regression analysis the correct input data must be available and the basic functional elements must be assumed. This models the system and allows for the regression coefficients to be solved [11, 15, 16], as indicated in Equation 2.1.

$$y(t) = b_0 + b_1 x_1(t) + \cdots + b_n x_n(t) + a(t) \tag{2.1}$$

where $y(t)$ is the predicted power; $x_1(t)$,$\cdots$,$x_n(t)$ is the explanatory variables corresponding to $y(t)$; $a(t)$ is a random variable constantly varying with a zero mean; and $b_0, b_1, \cdots, b_n$ are the regression coefficients [15].

2.3.2 Computational Intelligence Models

The emergence of computational intelligence has led to several methods being developed and applied to load forecasting.

Expert Systems

Expert systems are computer based algorithms that would require knowledge of an expert user to solve a well bounded problem domain [17]. Expert systems can be used in diagnostics, planning, control and it could even be used to forecast loads. This type of system replicates the knowledge and thought processes of a

'human expert' [18]. Expert systems have a configuration similar to that shown in Figure 2.2 [17, 19].

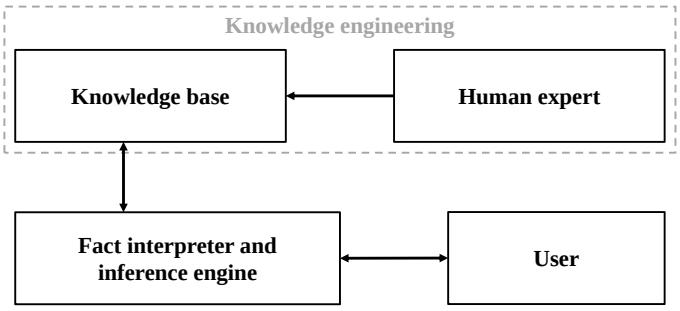


Figure 2.2: Overview of a typical expert system.

The knowledge base incorporates the data required to perform the process, such as power and weather for the load forecasting case. The inference engine uses a set of defined rules to search the knowledge base for the required data and interprets the results to produce the final output [19, 20]. The user interacts with the inference engine to define the search parameters and the human expert interacts with the knowledge base to produce the database that defines the expert system.

Artificial Neural Networks

Artificial neural networks, or more generally neural networks, are the computational simulation of the learning characteristics and pattern recognition of the human brain [21]. This ability to mimic the brain allows for the diverse application of neural networks to many different concepts such as system control, optical recognition and even load forecasting. Typical neural networks have a configuration similar to that shown in Figure 2.3 which is known as a 'percepton' artificial neural network [20, 22, 23].

The feed-forward network consists of 'neurons' (the quantity depending on the desired application) that are shown in Figure 2.4 [22]. These neurons will 'fire' if the bias of the input passes the threshold of the activation function. The most commonly used activation function is the sigmoid function [11, 22].

The neural network can be trained using various different algorithms including

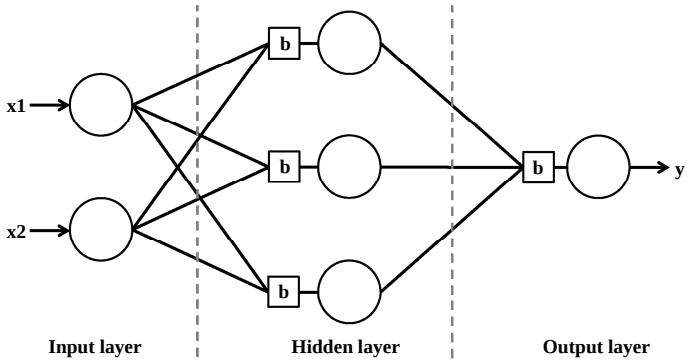


Figure 2.3: Overview of a general artificial neural network.

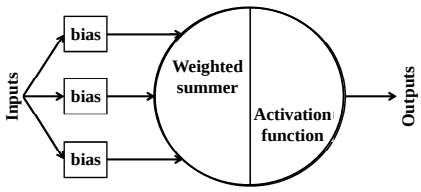


Figure 2.4: Simplified model of an artificial neuron.

the generalised delta rule, back propagation, genetic algorithms and other algorithms [23, 24, 25]. The training process physically changes the biasing threshold for each neuron such that the desired output is achieved for a known input-output pair.

Fuzzy Logic Systems

Fuzzy logic systems are generally applied to systems in which the mathematical model is non-linear or poorly understood. They typically have a configuration similar to that shown in Figure 2.5 [22], known as the Mamdani fuzzy logic system due to the linguistic nature of the rule-base [22, 26, 27, 28].

The inputs to the system are converted from crisp to fuzzy values in the fuzzification step. The fuzzy values are then interpreted by the fuzzy inference engine. The most common principle used for the fuzzy inference engine is the max-min inference process [22, 27]. When this is combined with the fuzzy rule-base ('Mamdani-style' rules because of the linguistic nature), the inputs can be quantified in a universe of discourse. During the defuzzification step the resulting fuzzy value is interpreted as a crisp value for each rule. The crisp values are then combined using the desired

equation thus yielding the final output of the fuzzy logic system [20, 22, 26, 27].

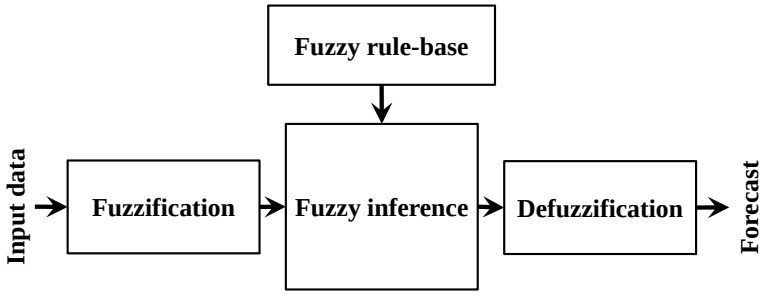


Figure 2.5: Overview of a Mamdani fuzzy logic system.

2.3.3 Combination of Methods

In addition to the above mentioned individual methods, there are hybrid methods that combine at least two of the methods mentioned to improve the results. Two common combinations of methods include the hybrid fuzzy-regression [29] and the hybrid fuzzy-neural networks [30, 31]. Further details can be found in the relevant papers written by Liang and Cheng [29]; Srinivasan, Chang and Tan [30]; Srinivasan and Lee [31] as well as Liao [32].

2.3.4 Comparison of Load Forecasting Methods

Several different implementations of each of the above mentioned methods have been compared in Table A.1 (in Appendix A), where the error that is presented is the mean absolute percentage error (MAPE), given in Equation 2.2.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|P_{forecast}(i) - P_{measure}(i)|}{P_{measure}(i)} \times 100 \tag{2.2}$$

Table 2.1 shows the average of the presented errors for each of implementations investigated. Each method of load forecasting had at least two different implementations to calculate the average errors.

Table 2.1: Average errors for each forecasting method based on the results presented in Table A.1.

Forecasting method	**Average error**
Time series analysis	1.11 %
Regression analysis	1.97 %
Expert systems	2.05 %
Artificial neural networks	2.11 %
Fuzzy logic systems	2.43 %
Combined methods	1.48 %

It can be inferred from Table 2.1 that the most effective method of load forecasting (based on the results presented in Table A.1) was the time series analysis. The method that yielded the highest average error was the fuzzy logic system. The trend observed was the more input data required yielded the lower MAPE across all forecasting methods.

The combination of methods reduced the forecasting MAPE. Incorporating the fuzzy logic system into the regression model, by Liang and Cheng [29], improved the MAPE by 13.00 % when compared to the regression analysis model by the same authors. The fuzzy-neural method, by Liao [32], improved the MAPE by 8.55 % when compared to the genetic algorithm optimised neural network created by the same author.

Due to these observations, it was proposed that fuzzy logic systems be investigated further and combine it with other methods, such as genetic algorithms, to illustrate the improved performance of the hybrid system.

Chapter 3

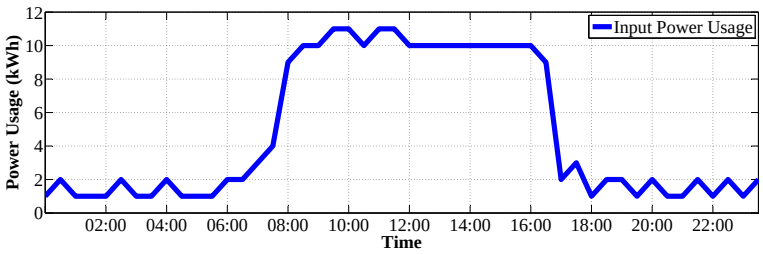
Development of the Load Forecasting Algorithm

Chapter Overview: *For this investigation into load forecasting, fuzzy logic systems were combined with genetic algorithms to provide the desired hybrid system. This chapter documents the development of the fuzzy logic systems for the load forecasting algorithm, as well as the development of the genetic algorithm to accurately parameterise the algorithm. The necessary assumptions and constraints are presented as well as the performance criteria. Important definitions to aid the understanding of this chapter are given in Appendix B.*

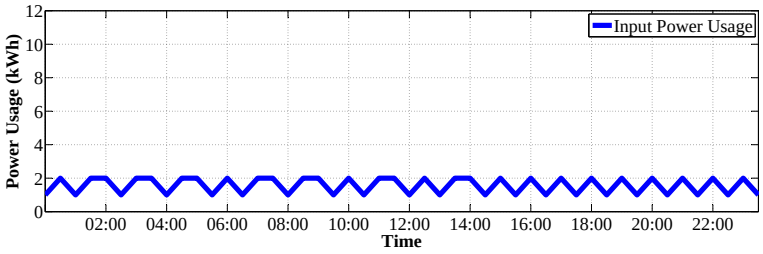
3.1 Assumptions and Constraints

Several assumptions and constraints were necessary in order to implement the load forecasting algorithm. These are:

- The original load profile that the system was designed for was a wing of a single floor of the Chamber of Mines engineering building at the University of the Witwatersrand (henceforth known as the University), with a profile illustrated in Figure 3.1.
- Power usage data (input and forecast) was normalised on 0 kWh - 12 kWh (due to the original load profile) to ensure that any load could be accommodated.



(a)



(b)

Figure 3.1: Sample load profile to illustrate the need to distinguish between (a): Week and (b): Weekend days in the load forecasting algorithm.

- 30 minute intervals were considered the normal period for forecasts (as per the South African power utility measurements, and measurements performed on the University campuses).
- 24 hours was considered the shortest time scale for the forecast in order to calculate performance.
- The maximum period for load forecasting was limited to a full week to reduce the required amount of input data.
- Week was defined as Monday to Friday, and weekend was defined as Saturday and Sunday.
- Public holidays were modelled as weekend days where necessary.
- Weather had minimal impact within the week long timescale and as such was neglected for the study.
- The fuzzy logic systems for each of the test cases were all the same before implementing the genetic algorithm to allow for comparison of the genetic algorithm performance.

These assumptions and constraints ensured the results obtained from the load forecasting algorithm would be accurate and valid, while remaining uncomplicated and prompt in operation.

3.2 Definition of Performance Criteria

Two criteria were defined to evaluate the performance of the load forecasting algorithm. They were:

1. The difference between the forecast peak load and the measured peak load for a 24 hour period (or peak energy error),

$$E_{peak} = \frac{|\max(P_{forecast}) - \max(P_{measured})|}{\max(P_{measured})} \times 100 \tag{3.1}$$

and

2. The difference between the total energy required for the forecast load and the measured load in a 24 hour period (or total energy error),

$$E_{total} = \int_1^n \frac{|P_{forecast}(t) - P_{measured}(t)|}{P_{measured}(t)} dt \times 100 \tag{3.2}$$

where: $P_{forecast}$ is the forecast power usage,
$P_{measured}$ is the measured power usage,
n is the maximum number of terms in the forecast period.

It should be noted that the integral in Equation 3.2 was performed using trapezoidal numerical integration to give a close approximation to the true value.

These performance criteria were chosen since they are some of the most important factors when considering energy management [10]. If the peak load and the total energy usage in a 24 hour period can be forecast then the user can implement methods of reducing it, or the power producer can plan accordingly to ensure that the required peak is available. This would lead to a greater awareness of the power usage and possibly improve bill management for the user. The defined performance criteria utilised a backwards comparison approach, meaning the performance could only be established for the week-ahead forecast once the week had been completed.

In order to evaluate the effectiveness of the genetic algorithm, a simple equation was defined to show the ratio between the result before and after the genetic algorithm was implemented. This ratio can also be considered the improvement factor due to the genetic algorithm. This is illustrated as:

$$GA_{improve} = \frac{E_{before}}{E_{after}} \tag{3.3}$$

where: E_{before} is the error before the genetic algorithm is implemented,
E_{after} is the error after the genetic algorithm is implemented.

3.3 Outline of the Load Forecasting Algorithm

The load forecasting algorithm comprised of two fuzzy logic systems performing the load forecasting, one for weekday forecasts and the other for weekend forecasts due to differences in the observed trends for the types of day as illustrated in Figure 3.1. The inputs to the algorithm were the time of day and the historic power usage. A third input was the day of the week, however it would not impact the fuzzy logic systems. It rather selected which of the fuzzy logic systems to use whether the day was during the week or weekend. A genetic algorithm was developed to adjust the parameters for the fuzzy logic membership functions and rules to maximise the performance of the load forecasting algorithm. The genetic algorithm was not implemented in real-time, but rather once a week (every seven days) due to computational requirements. For each new week the genetic algorithm would need to be reimplemented to maintain a high level of accuracy. The logical flow diagram of the algorithm is illustrated in Figure 3.2, where the output of the load forecasting algorithm is shown in blue.

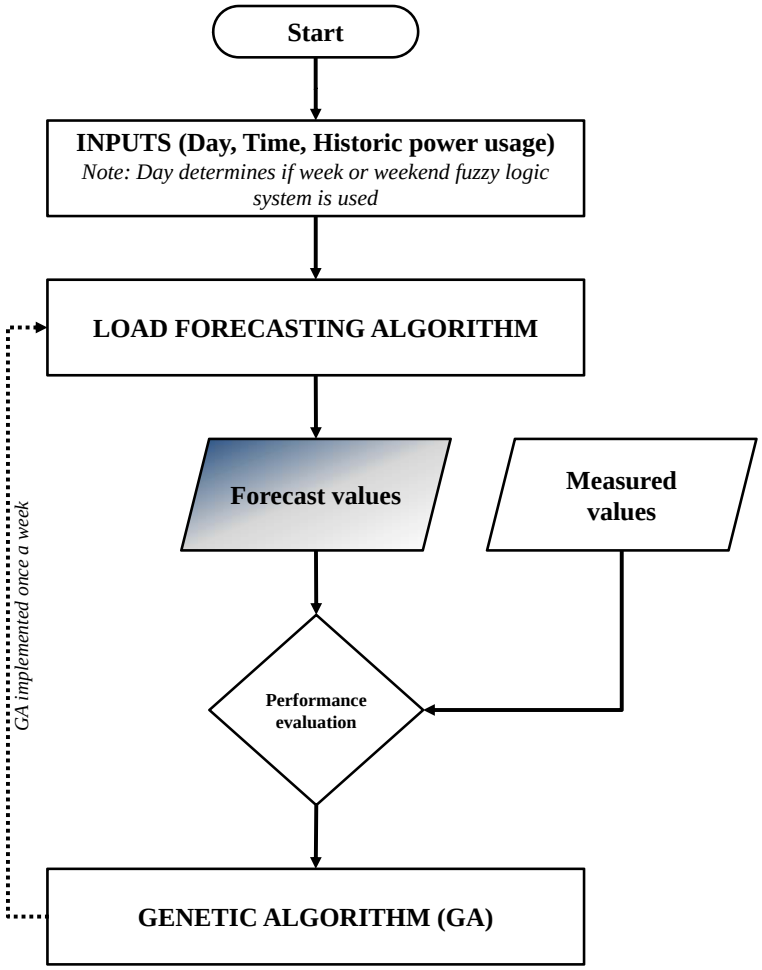


Figure 3.2: Flow diagram showing the processes for the load forecasting algorithm.

3.4 The Fuzzy Logic Systems

The fuzzy logic systems that were designed followed the structure depicted in Figure 3.3. Both fuzzy logic systems were designed to be the same structure, and only the parameters would differ between week days and weekend days.

The input and output sets of the fuzzy logic system used a symmetrical Gaussian distribution for each of the *membership functions*, seen in Equation 3.4 with μ and σ being some of the variables that were parameterised during the genetic algorithm parameterisation loop. The continuous nature of the Gaussian distribution allows for enhanced optimisation since there are no discontinuous points at which the

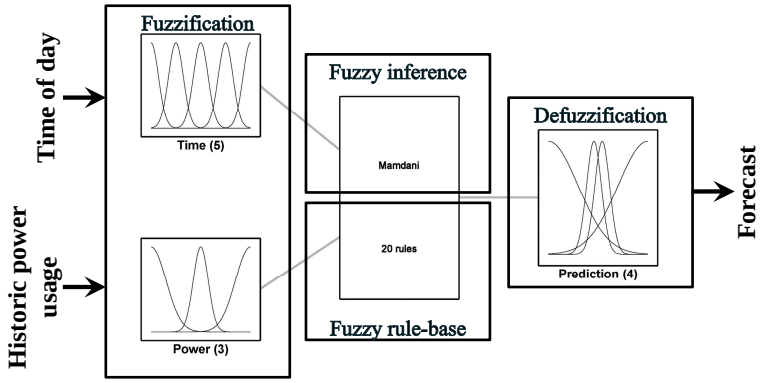


Figure 3.3: Overview of the fuzzy logic system used in the load forecasting algorithm.

optimisation could fail.

$$f(x) = e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} \tag{3.4}$$

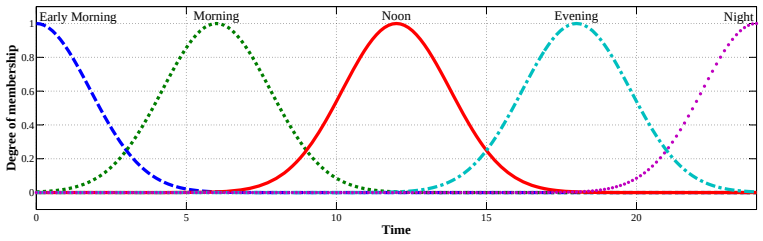
where: μ is the mean of the distribution,
σ is the standard deviation of the distribution.

The positions (μ) and widths (σ) of each of the membership functions for the input and output fuzzy sets were selected on a trial-and-error basis to yield the most accurate results.

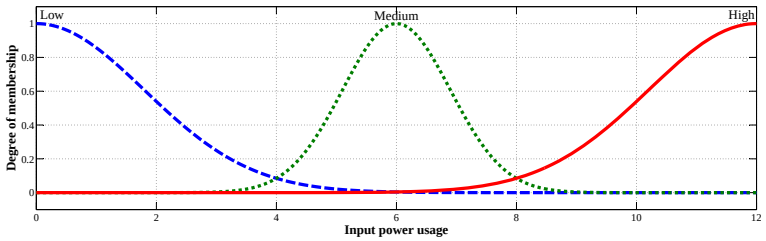
3.4.1 Fuzzification Process

Two input sets were used in the fuzzy logic system to *fuzzify* the *crisp input values*. The required inputs were the time of day, shown in Figure 3.4(a), and the historic power usage, shown in Figure 3.4(b).

The crisp values were passed to the fuzzy logic systems, and converted to *fuzzy values* through the fuzzification process. The crisp value (on the x-axis) was assigned a *degree of membership* (on the y-axis) based on which membership function curve it corresponded to.



(a)



(b)

Figure 3.4: Input fuzzy sets for (a): Time and (b): Historic power usage used in the load forecasting algorithm

3.4.2 Fuzzy Inference Engine

Fuzzy inference was established by using a linguistic *fuzzy rule-base* as well as the *max-min inference* process. The fuzzy rule-base consisted of 20 Mamdani-type linguistic rules, shown in Table 3.1. The weight of each rule, being a variable for parameterisation using the genetic algorithm, was initially set to one. This implied that each rule contributes to the final output equally. The rules were derived using *a priori* knowledge of the behaviour of the system, based on the load profile shown in Figure 3.1.

3.4.3 Defuzzification

The outputs of the fuzzy inference engine were combined with a single crisp output set, shown in Figure 3.5, to *defuzzify* the fuzzy values. This yielded the desired crisp value, being the forecast load profile.

Table 3.1: Mamdani-style fuzzy rule-base used in the load forecasting algorithm.

1	IF	Time is Early Morning	AND	Power is Low	THEN	Forecast is Very Low
2	IF	Time is Early Morning	AND	Power is Medium	THEN	Forecast is Low
3	IF	Time is Early Morning	AND	Power is Medium	THEN	Forecast is Medium
4	IF	Time is Early Morning	AND	Power is High	THEN	Forecast is High
5	IF	Time is Morning	AND	Power is Low	THEN	Forecast is Very Low
6	IF	Time is Morning	AND	Power is Medium	THEN	Forecast is Low
7	IF	Time is Morning	AND	Power is Medium	THEN	Forecast is Medium
8	IF	Time is Morning	AND	Power is High	THEN	Forecast is High
9	IF	Time is Noon	AND	Power is Low	THEN	Forecast is Very Low
10	IF	Time is Noon	AND	Power is Medium	THEN	Forecast is Low
11	IF	Time is Noon	AND	Power is Medium	THEN	Forecast is Medium
12	IF	Time is Noon	AND	Power is High	THEN	Forecast is High
13	IF	Time is Evening	AND	Power is Low	THEN	Forecast is Very Low
14	IF	Time is Evening	AND	Power is Medium	THEN	Forecast is Low
15	IF	Time is Evening	AND	Power is Medium	THEN	Forecast is Medium
16	IF	Time is Evening	AND	Power is High	THEN	Forecast is High
17	IF	Time is Night	AND	Power is Low	THEN	Forecast is Very Low
18	IF	Time is Night	AND	Power is Medium	THEN	Forecast is Low
19	IF	Time is Night	AND	Power is Medium	THEN	Forecast is Medium
20	IF	Time is Night	AND	Power is High	THEN	Forecast is High

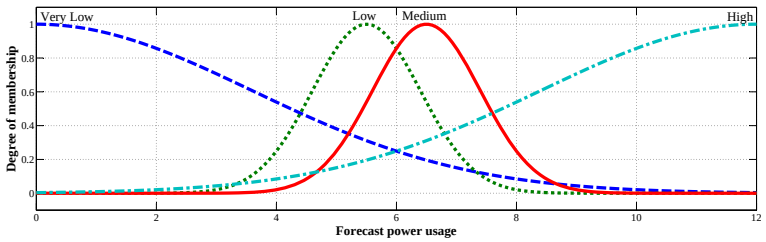


Figure 3.5: Output fuzzy set used in the forecaster.

The *mean of maximum* calculation was used for defuzzification. An example of this is shown in Figure 3.6. This calculation determines the mean value of all the possible crisp values that correspond to the maximum output fuzzy value, thus yielding a single predicted value.

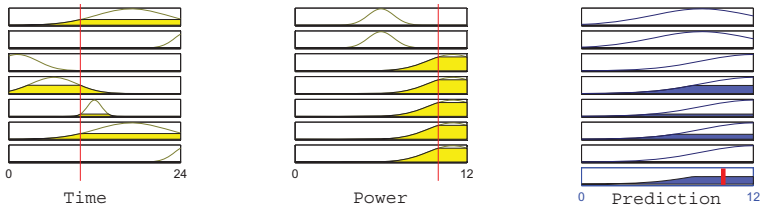


Figure 3.6: Illustration of the mean of maximum defuzzification calculation. Time = 10h00; Historic power usage = 10 kWh; Predicted power usage (after defuzzification) = 9.9 kWh.

3.5 The Genetic Algorithm

The genetic algorithm developed to optimise the load forecasting algorithm, similar to the conventional genetic algorithm shown in Figure 3.7 [33, 34, 35], consisted of 44 variables in the *population*. The variables were the μ and σ for each of the fuzzy membership functions in all of the input and output fuzzy sets, as well as the weight of each rule in the fuzzy rule-base. The lower and upper bounds for each of the variables were defined to ensure the genetic algorithm could converge to a final solution that was within the fuzzy problem-space (or *universe of discourse*).

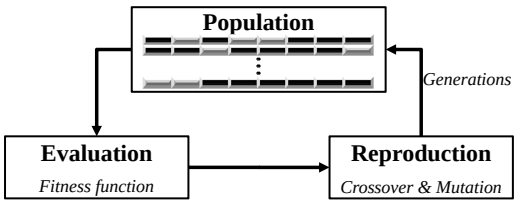


Figure 3.7: Overview of a conventional genetic algorithm.

3.5.1 Background

Genetic algorithms are search procedures that serve as an effective method of simulating evolution and natural selection. Due to this feature, genetic algorithms have found significant use in the field of optimisation [20, 36]. The most notable work on genetic algorithms was that by Holland (1975) [33] advancing the theoretical knowledge as well as Goldberg (1989) [34] who developed innovative applications.

The genetic algorithm functions by creating an initial population of *chromosomes* (or solutions), based on the number of variables and the desired population size. Each solution is evaluated and ranked according to a defined *fitness function*. The 'fittest' solutions are then selected to breed the next generation of solutions, including evolutionary effects such as gene *crossover* and *mutation* [20, 22, 35, 37]. The flow process for a conventional genetic algorithm is shown in Figure 3.7.

3.5.2 Chromosome Representation

The chromosome of the genetic algorithm was the initial 'guess' at the variables to be solved. The initial guesses as to the initial chromosome for a week day and a weekend day can be seen in lines 3 and 4 of Table 3.2, Table 3.3 and Table 3.4. These values were determined using *a priori* knowledge of the system. Thus the fuzzy logic systems were 'tuned' until accurate results were obtained. Lines 1 and 2 in Table 3.2, Table 3.3 and Table 3.4 indicate the lower and upper boundary of the search space for the chromosome such that the fuzzy logic systems were contained within the defined universe of discourse.

Table 3.2: Chromosome information of the genetic algorithm for the fuzzy input membership functions.

	Time		**Power**	
	μ	σ	μ	σ
1. Lower	[0 0 0 0 0]	[0.01 0.01 0.01 0.01 0.01]	[0 0 0]	[0.01 0.01 0.01]
2. Upper	[25 25 25 25]	[5 5 5 5 5]	[13 13 13]	[4 4 4]
3. Week	[0 6 12 18 24]	[1.8 1.8 1.8 1.8 1.8]	[0 6 12]	[1.8 0.9 1.8]
4. Weekend	[0 6 12 18 24]	[4.2 2.0 3.2 2.0 3.9]	[0 6 12]	[2.5 2.5 2.5]

Table 3.3: Chromosome information of the genetic algorithm for the fuzzy output membership funcitons.

	Forecast	
	μ	σ
1. Lower	[0 0 0 0]	[0.01 0.01 0.01 0.01]
2. Upper	[13 13 13 13]	[4 4 4 4]
3. Week	[0 5.5 6.5 12]	[3.6 0.9 0.9 3.6]
4. Weekend	[0 5.5 6.5 12]	[3.8 1.8 1.8 3.8]

Table 3.4: Chromosome information of the genetic algorithm for the fuzzy rule set weights.

	Fuzzy Rule Weights
1. Lower	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
2. Upper	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
3. Week	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
4. Weekend	[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

The population size was set to 120 to ensure greater variability in the population. The maximum number of *generations* was set to 200 such that adequate cycles were completed to converge on the global solution.

3.5.3 Objective/Fitness Functions

The fitness function defined for the genetic algorithm was the average of the two performance criteria, as indicated in Equation 3.5. This ensured that the average of the two errors was reduced to a minimum, since the MATLAB® implementation of the fitness function sought the global minimum solution.

$$F_{fit} = \frac{E_{peak} + E_{total}}{2} \tag{3.5}$$

Fitness scaling was selected to be *rank scaling*. This improved the probability of the fittest of the population to progress to the next generation. The stopping criteria for the genetic algorithm was set to a *fitness function tolerance* of 1×10^{-9}.

3.5.4 Reproduction

The method of genetic *reproduction* was implemented as *stochastic uniform* function. The *parents* resulting from this process underwent the two major reproduction processes, crossover and mutation. An elitist method was adopted to ensure at least four of the weaker *offspring* continued to the following generation.

Crossover

The percentage of the population affected by crossover was set to 0.8, meaning that 80 % of the next generation was created due to crossover. The crossover function used was the *scattered function* to ensure a diverse population of offspring.

Mutation

Since the percentage of the population being affected by crossover was 0.8, the percentage affected by mutation was thus 0.2, meaning mutation accounted for 20 % of the following generation. The mutation function used was the *adaptive feasible function*. This allowed for a potentially weak chromosome to become stronger and improve the fitness of the population.

Generations

Once the new population had been completely created, the next generation began. The offspring became the parents in the new fitness function test and the process repeated until the maximum number of generations was reached (set to 200 generations for this study) or the fitness function tolerance was fulfilled (defined as 1×10^{-9} for this study).

3.6 Structure of the Load Forecasting Algorithm

The algorithm was developed and implemented using MATLAB® toolboxes. The fuzzy logic system was created using the fuzzy logic toolbox and the genetic algorithm was created using the optimisation toolbox. The load forecasting algorithm, genetic algorithm and performance criteria calculation processes are shown in Figure 3.8.

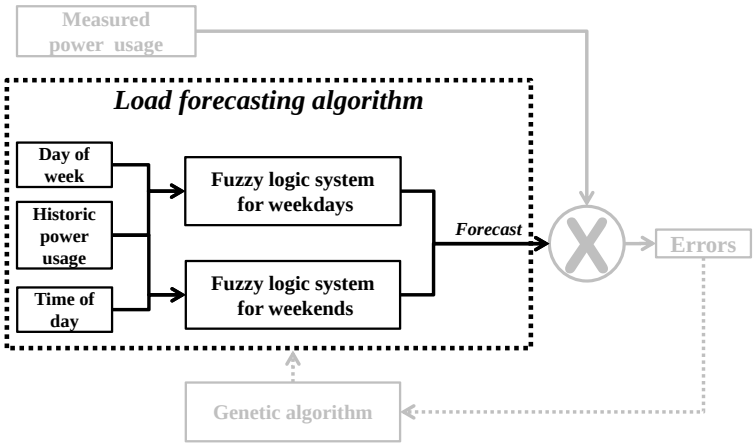


Figure 3.8: Structure of the load forecasting algorithm (in black) and the accompanying performance evaluation and optimisation processes (in grey).

Chapter 4

Algorithm Testing, Results and Analysis

Chapter Overview: *This chapter presents the results of different tests performed to determine the input data requirements to yield the most accurate results, and for each of the case studies described. The case studies were performed before and after the genetic algorithm was implemented to observe the effects of the genetic algorithm on the fuzzy logic systems. Five different load profiles were tested. The loads were (from largest to smallest): the Eastern Cape Province in South Africa; as well as the East Campus at the University of the Witwatersrand, Johannesburg, South Africa (henceforth known as the University), Barnato Hall student residence at the University, Chamber of Mines engineering building at the University and a single plug point with a variable load. Graphical results are only generated for after the genetic algorithm implementation for illustrative purposes. The MATLAB® code used to develop the load forecasting algorithm is presented in Appendix C. Additional information pertaining to the case studies is given in Appendix D.*

4.1 Preliminary Algorithm Testing to Determine Input Requirements

The desired amount of input data required was determined by using a varying number of weeks' historic power usage data as the input to the load forecasting algorithm. Three options were considered: using one, two or three weeks' data prior to the test as the input to the algorithm. The input for the two weeks and three weeks options was the average of the data from each of the two and three weeks prior to the test respectively. The results for each of these tests are shown in Table 4.1.

Table 4.1: Error calculations for the varying inputs test before the genetic algorithm was implemented.

Quantity of input data	E_{peak} (%)	E_{total} (%)	**Average (%)**
One weeks data	0.36	3.28	**1.82**
Two weeks data	4.55	4.26	**4.40**
Three weeks data	2.91	5.79	**4.35**

The results indicate that using data from a week prior to the test yielded the most accurate prediction capabilities with an average error of 1.82 % determined from E_{peak} and E_{total}. This was better than using either of the two other proposed inputs by a factor of approximately 2.4. Therefore using data from one week prior to the test was used for the detailed case studies.

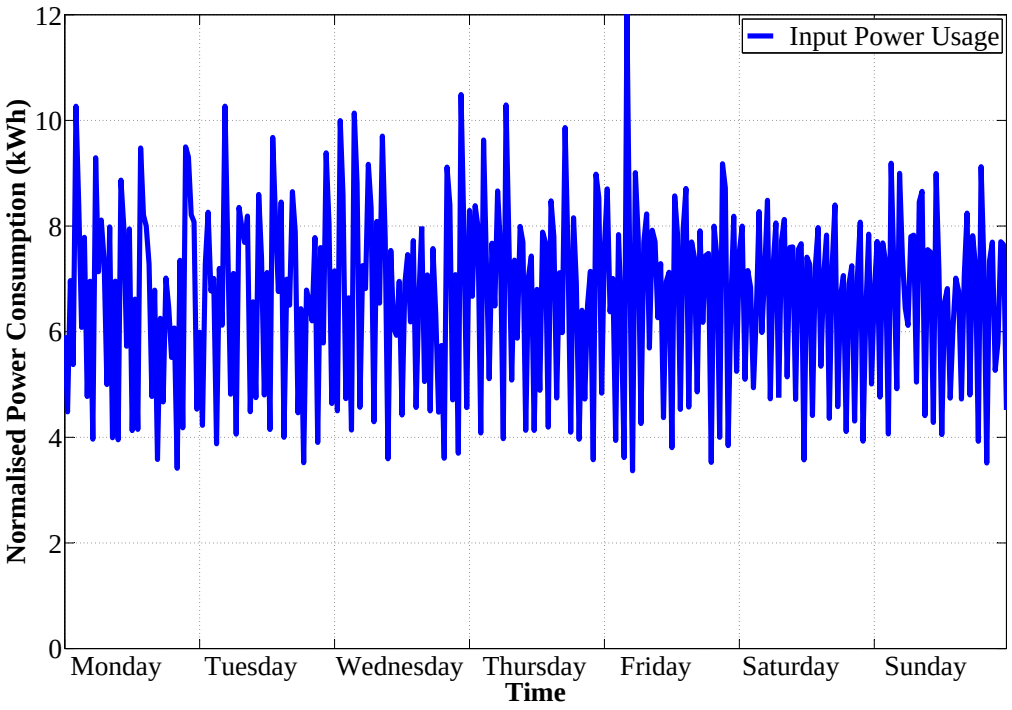
4.2 Case 1 - Eastern Cape Province in South Africa

The energy usage data for this case was obtained from the South African power producer. The area of the load profile, relative to the whole country, is shown in Figure D.1 in Appendix D. The results for the load forecasting test on this load can be seen in Table 4.2. A graphical summary of the week-long forecast can be seen in Figure 4.1(b). Individual daily summaries to better illustrate the load forecasting capabilities can be seen in Figure D.2 to Figure D.8 in Appendix D.

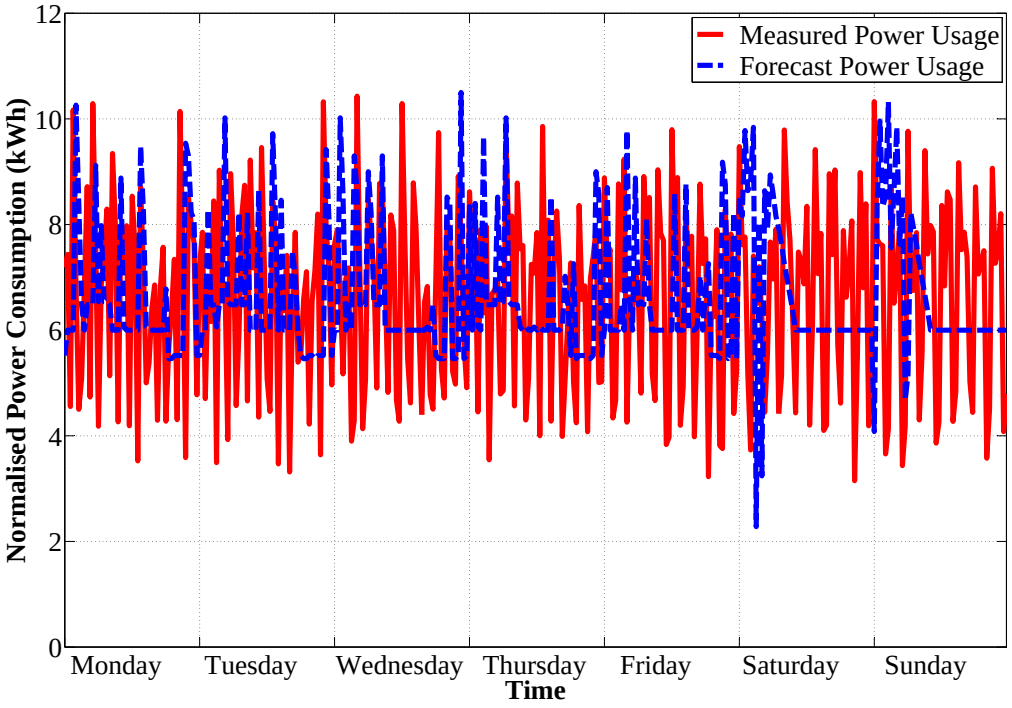
Table 4.2: Load forecasting results for the Eastern Cape Province, before and after the genetic algorithm (GA) was implemented.

	Before GA		**After GA**	
	E_{peak} (%)	E_{total} (%)	E_{peak} (%)	E_{total} (%)
Monday	2.62	3.99	0.28	0.01
Tuesday	0.03	2.58	2.94	0.00
Wednesday	0.68	0.43	0.68	0.50
Thursday	4.71	1.27	1.66	0.01
Friday	2.29	5.97	0.16	0.01
Week Average	*2.06*	*2.85*	*1.14*	*0.11*
Saturday	38.72	9.91	0.50	0.01
Sunday	4.13	8.17	0.07	0.00
Weekend Average	*21.43*	*9.04*	*0.29*	*0.01*
AVERAGE	**7.60**	**4.62**	**0.90**	**0.08**

The average peak error over the period of a week was 0.90 % and the average total energy error was 0.08 % for the forecast load after the genetic algorithm was implemented. When compared to the results from before the genetic algorithm implementation (E_{peak} - 7.60 % and E_{total} - 4.62 %), an improvement due to the genetic algorithm by factors of 8.45 and 60.26 for E_{peak} and E_{total} respectively was observed.



(a)



(b)

Figure 4.1: (a): Input to the load forecasting algorithm and (b): Full week forecast for the Eastern Cape Province in South Africa.

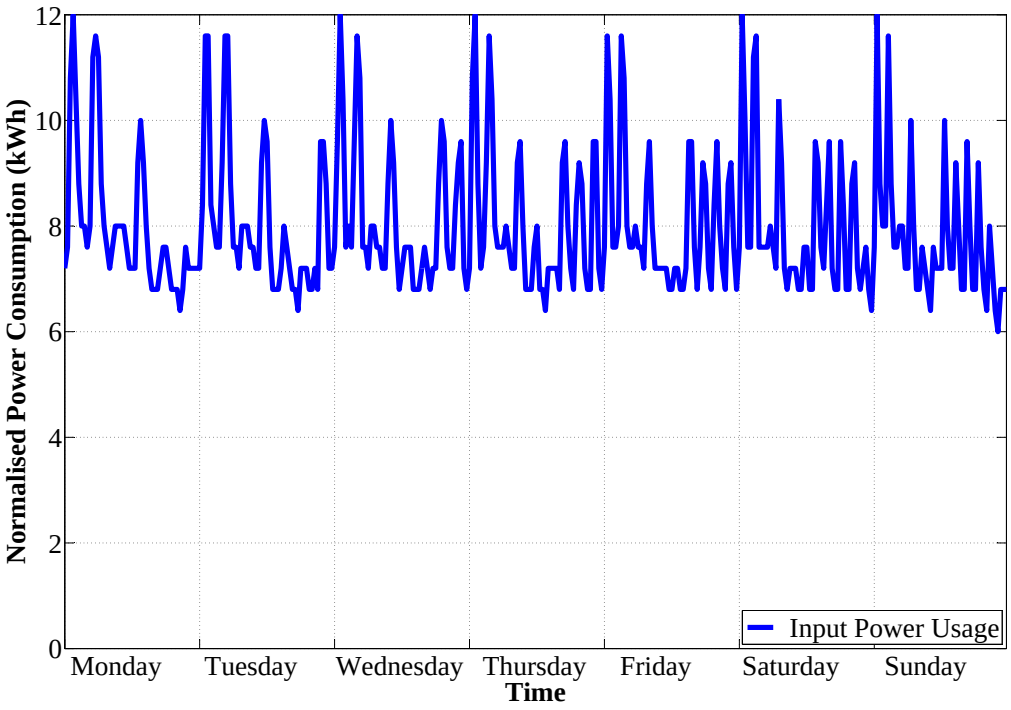
4.3 Case 2 - East Campus at the University

Load profile data was acquired from the installed power meters for the main feed of East Campus at the University (shown in Figure D.9, Appendix D). The results for the load forecasting test on this load can be seen in Table 4.3. A graphical summary of the week-long forecast can be seen in Figure 4.2(b). Individual daily summaries to better illustrate the load forecasting capabilities can be seen in Figure D.10 to Figure D.16 in Appendix D.

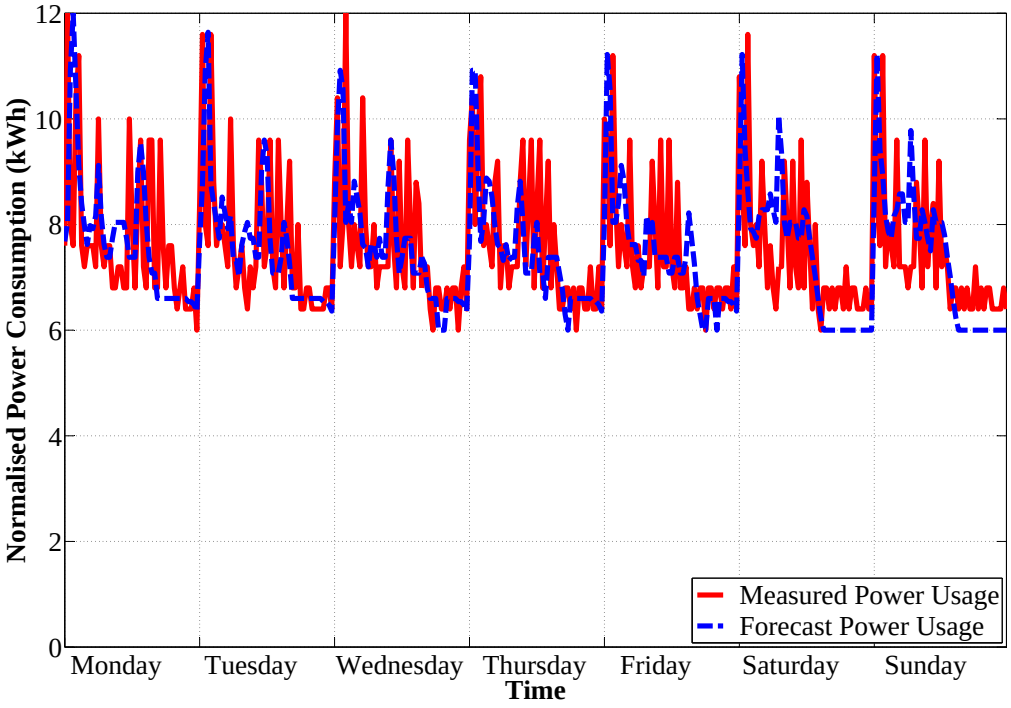
Table 4.3: Load forecasting results for East Campus at the University, before and after the genetic algorithm (GA) was implemented.

	Before GA		**After GA**	
	E_{peak} (%)	E_{total} (%)	E_{peak} (%)	E_{total} (%)
Monday	4.00	5.74	0.00	0.01
Tuesday	4.83	4.77	0.34	0.01
Wednesday	8.00	3.61	9.00	0.00
Thursday	2.22	4.49	1.11	2.25
Friday	2.86	2.86	0.18	0.00
Week Average	*4.38*	*4.29*	*2.13*	*0.45*
Saturday	1.90	4.40	3.28	0.00
Sunday	5.54	9.11	0.18	0.01
Weekend Average	*3.72*	*6.76*	*1.73*	*0.00*
AVERAGE	**4.19**	**5.00**	**2.01**	**0.32**

The average peak error over the period of a week was 2.01 % and the average total energy error was 0.32 % for the forecast load after the genetic algorithm was implemented. When results from the fuzzy logic implementation alone (E_{peak} - 4.19 % and E_{total} - 5.00 %) were compared to the fuzzy logic and genetic algorithm combination, an improvement due to the genetic algorithm by factors of 2.08 and 15.40 for E_{peak} and E_{total} respectively was seen.



(a)



(b)

Figure 4.2: (a): Input to the load forecasting algorithm and (b): Full week forecast for East Campus at the University.

4.4 Case 3 - Barnato Hall Student Residence at the University

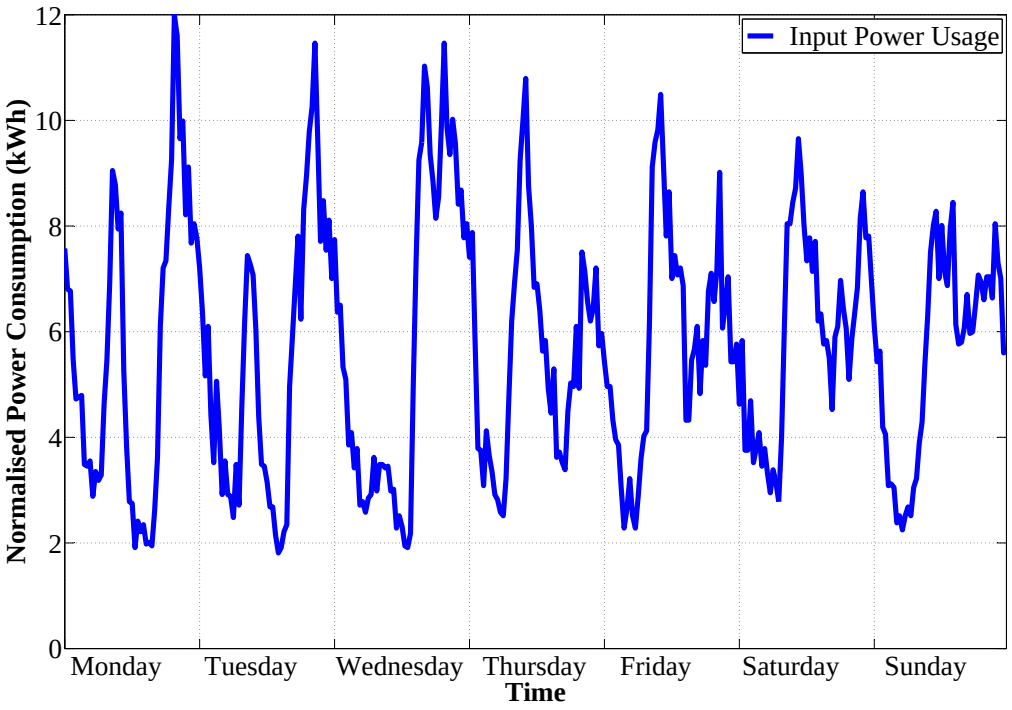
Load profile data was acquired from the installed power meters for the main feed of the Barnato Hall student residence at the University (shown in Figure D.17, Appendix D). Barnato Hall provides housing for 370 students each with individual rooms [38].

The results for the load forecasting test on this load can be seen in Table 4.4. A graphical summary of the week-long forecast can be seen in Figure 4.3(b). Individual daily summaries to better illustrate the load forecasting capabilities can be seen in Figure D.18 to Figure D.24 in Appendix D.

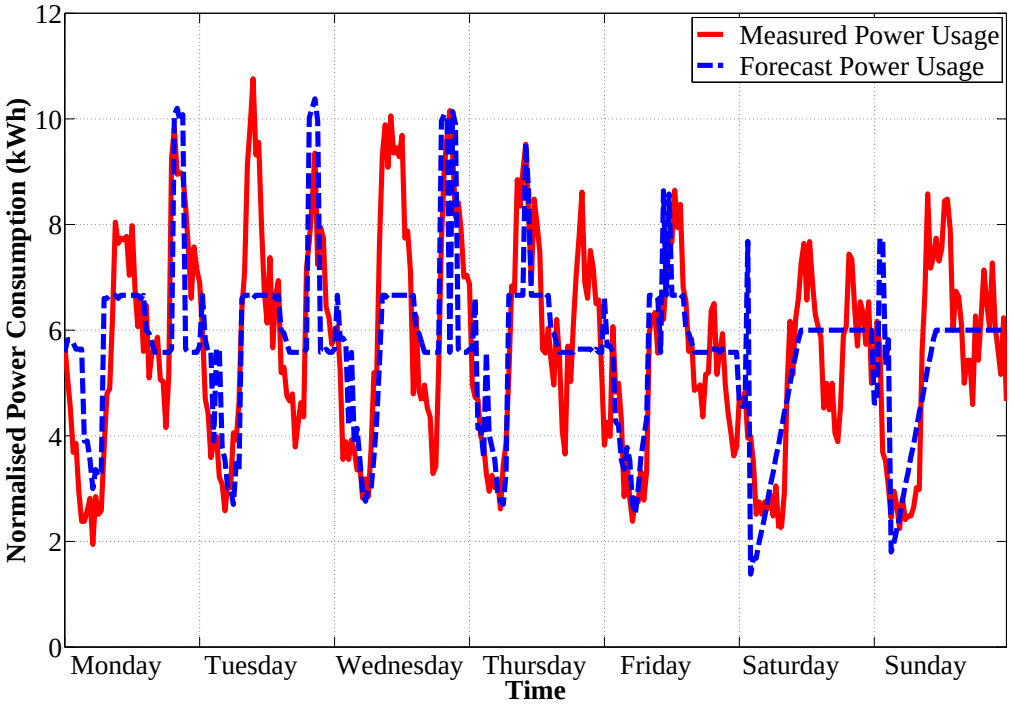
Table 4.4: Load forecasting results for the Barnato Hall student residence at the University, before and after the genetic algorithm (GA) was implemented.

	Before GA		**After GA**	
	E_{peak} (%)	E_{total} (%)	E_{peak} (%)	E_{total} (%)
Monday	6.91	3.64	3.86	5.21
Tuesday	6.88	10.72	3.53	0.04
Wednesday	3.38	9.47	0.16	5.71
Thursday	5.26	6.25	0.21	4.09
Friday	15.86	12.75	0.09	5.40
Week Average	*7.66*	*8.57*	*1.57*	*4.09*
Saturday	33.66	19.26	0.05	0.57
Sunday	30.08	1.61	9.80	2.60
Weekend Average	*31.87*	*10.44*	*4.93*	*1.58*
AVERAGE	**14.58**	**9.10**	**2.53**	**3.37**

The average peak error over the period of a week was 2.53 % and the average total energy error was 3.37 % for the forecast load after the genetic algorithm was implemented. When compared to the results from before the genetic algorithm implementation (E_{peak} - 14.58 % and E_{total} - 9.10 %), an improvement due to the genetic algorithm by factors of 5.76 and 2.70 for E_{peak} and E_{total} respectively was observed.



(a)



(b)

Figure 4.3: (a): Input to the load forecasting algorithm and (b): Full week forecast for the Barnato Hall student residence at the University.

4.5 Case 4 - Chamber of Mines Building at the University

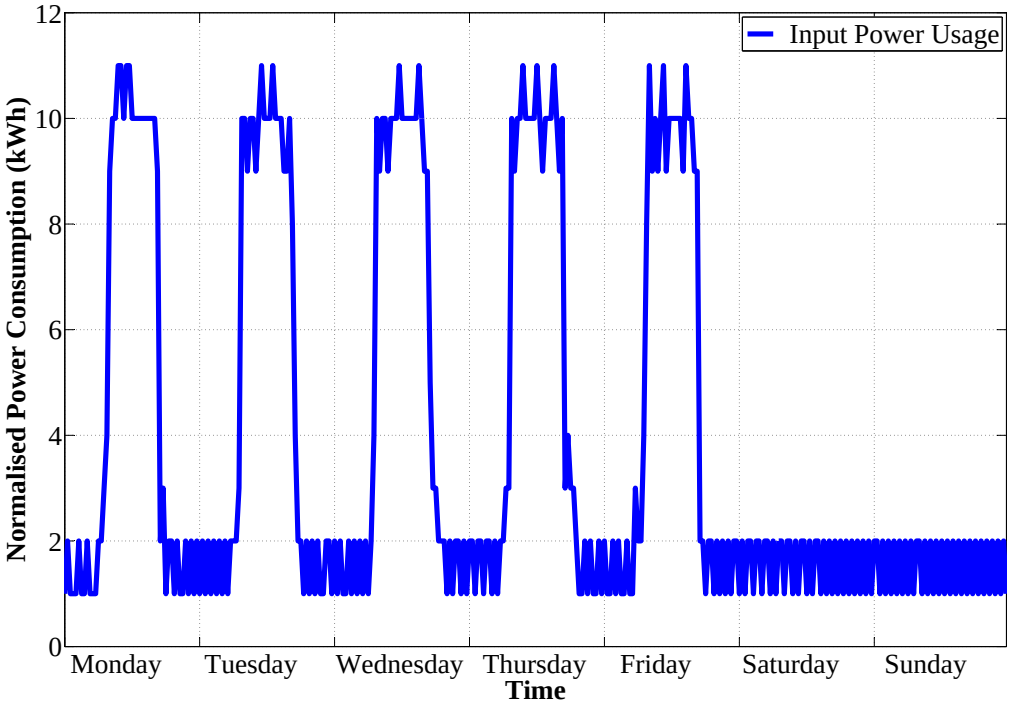
Load profile data was acquired from the installed power meters for the Chamber of Mines engineering building, on the West Campus at the University, shown in Figure D.25 in Appendix D. A wing of a single floor of the building, that was designated office area, was used for the test case to illustrate the load forecasting capability for an office environment.

The results for the load forecasting test on this load can be seen in Table 4.5. A graphical summary of the week-long forecast can be seen in Figure 4.4(b). Individual daily summaries to better illustrate the load forecasting capabilities can be seen in Figure D.26 to Figure D.32 in Appendix D.

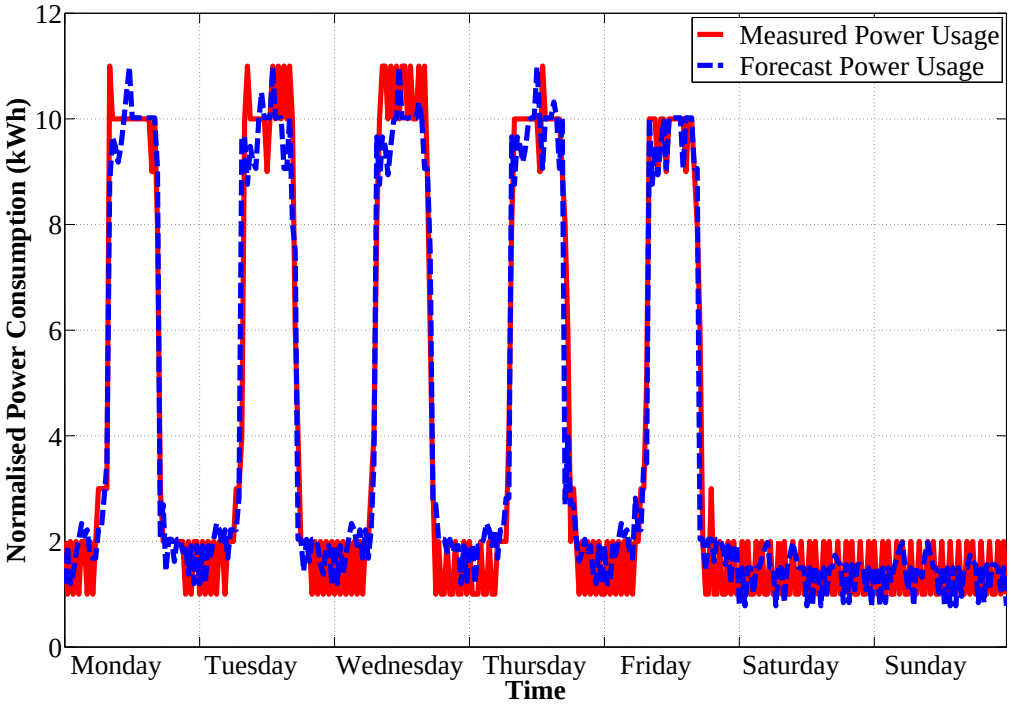
Table 4.5: Load forecasting results for a wing of a single story of the Chamber of Mines engineering building at the University, before and after the genetic algorithm (GA) was implemented.

	Before GA		**After GA**	
	E_{peak} (%)	E_{total} (%)	E_{peak} (%)	E_{total} (%)
Monday	0.36	0.42	0.18	0.10
Tuesday	0.36	0.53	0.18	0.25
Wednesday	0.36	0.00	0.18	0.02
Thursday	0.36	1.26	0.18	0.03
Friday	5.00	3.34	0.20	0.95
Week Average	*1.29*	*1.11*	*0.19*	*0.27*
Saturday	2.00	0.12	1.00	0.03
Sunday	2.00	0.98	1.00	0.00
Weekend Average	*2.00*	*0.55*	*1.00*	*0.02*
AVERAGE	**1.49**	**0.95**	**0.42**	**0.20**

The average peak error over the period of a week was 0.42 % and the average total energy error was 0.20 % for the forecast load after the genetic algorithm was implemented. The genetic algorithm exhibited improvements to the original fuzzy logic load forecasting algorithm (E_{peak} - 1.49 % and E_{total} - 0.95 %) by factors of 3.57 and 4.79 for E_{peak} and E_{total} respectively.



(a)



(b)

Figure 4.4: (a): Input to the load forecasting algorithm and (b): Full week forecast for the Chamber of Mines engineering building at the University.

4.6 Case 5 - Single Plug Point with a Variable Load

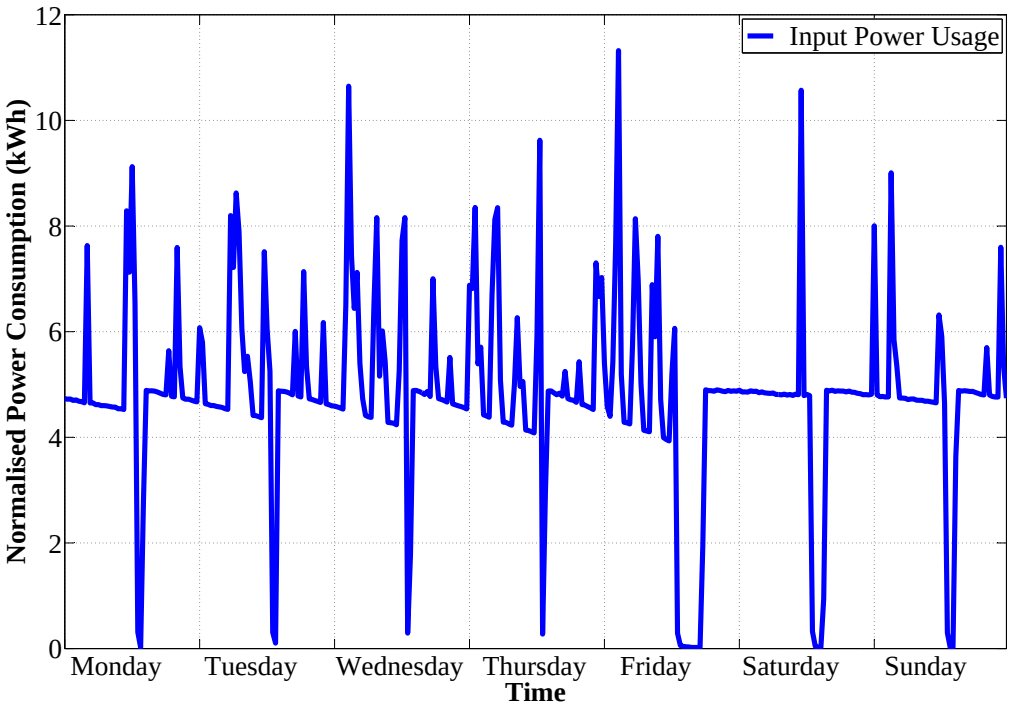
Load profile data was measured using a digital power meter. A single plug point was used, with the load being a computer and the power meter (to provide the base load) and a coffee machine as the variable (non-stochastic) load. The load could be said to be non-stochastic since the use of the coffee machine varied from day to day as well as at different times during the day.

The results for the load forecasting test on this load can be seen in Table 4.6. A graphical summary of the week-long forecast can be seen in Figure 4.5(b). Individual daily summaries to better illustrate the load forecasting capabilities can be seen in Figure D.33 to Figure D.39 in Appendix D.

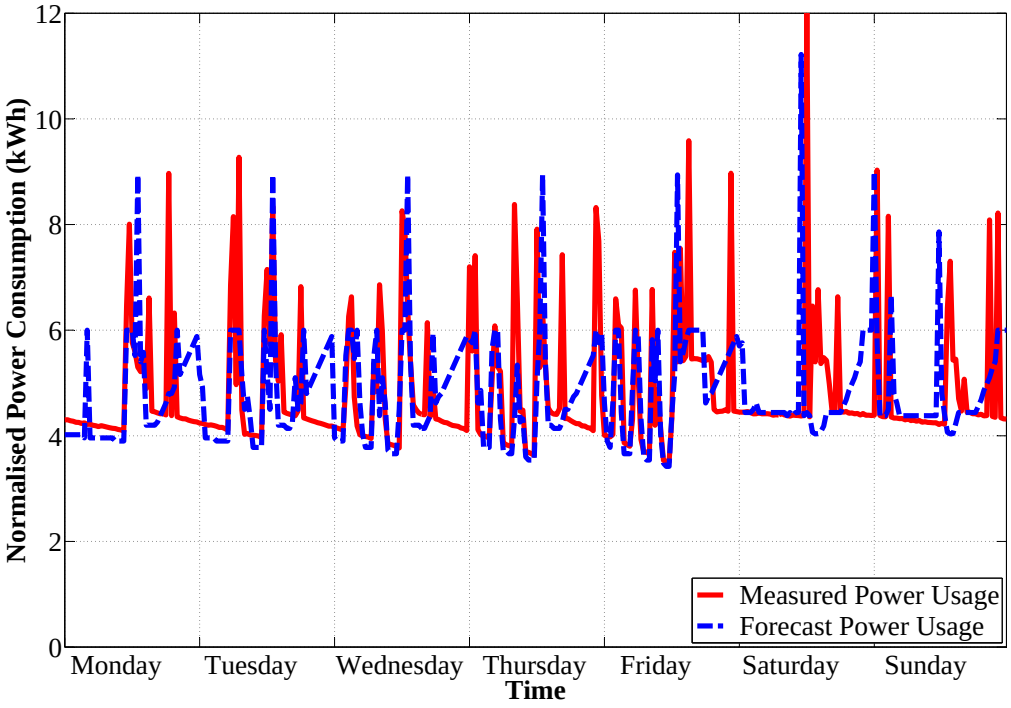
Table 4.6: Load forecasting results for a single plug point with a variable load, before and after the genetic algorithm (GA) was implemented.

	Before GA		**After GA**	
	E_{peak} (%)	E_{total} (%)	E_{peak} (%)	E_{total} (%)
Monday	2.37	23.64	0.31	0.30
Tuesday	6.82	21.84	3.59	0.49
Wednesday	15.51	26.49	8.24	3.28
Thursday	15.29	22.83	6.69	3.15
Friday	0.48	1.23	6.74	0.45
Week Average	*8.09*	*19.21*	*5.11*	*1.53*
Saturday	9.00	16.06	6.50	0.63
Sunday	8.30	18.50	0.33	1.06
Weekend Average	*8.65*	*17.28*	*3.42*	*0.85*
AVERAGE	**8.25**	**18.66**	**4.63**	**1.34**

The average peak error over the period of a week was 4.63 % and the average total energy error was 1.34 % for the forecast load after the genetic algorithm was implemented. When this result was compared to the results from before the genetic algorithm implementation (E_{peak} - 8.25 % and E_{total} - 18.66 %), an improvement due to the genetic algorithm by factors of 1.78 and 13.96 for E_{peak} and E_{total} respectively was shown.



(a)



(b)

Figure 4.5: (a): Input to the load forecasting algorithm and (b): Full week forecast for a single plug point with a variable load.

4.7 Analysis of Results

A summary of the full week forecasts for each of the case studies is shown in Table 4.7. It illustrates a consolidated view of the performance criteria (before and after implementation of the genetic algorithm) as well as the genetic algorithm improvement factors.

Table 4.7: Overall average results for the week-ahead forecast for each of the five case studies, showing the results for before and after the genetic algorithm (GA) was implemented as well as the GA improvement factor.

	Before GA		**After GA**		$GA_{improve}$	
	E_{peak} (%)	E_{total} (%)	E_{peak} (%)	E_{total} (%)	E_{peak}	E_{total}
Case 1	7.60	4.62	0.90	0.08	8.44	57.75
Case 2	4.19	5.00	2.01	0.32	2.08	15.63
Case 3	14.58	9.10	2.53	3.37	5.76	2.70
Case 4	1.49	0.95	0.42	0.20	3.55	4.75
Case 5	8.25	18.66	4.63	1.34	1.78	13.93
AVERAGE	**7.22**	**7.67**	**2.10**	**1.06**	**4.32**	**18.95**

As can be seen in Table 4.7 above, the average peak energy error before the genetic algorithm was implemented was 7.22 %. This was reduced to 2.10 % after implementing the genetic algorithm, indicating an improvement factor of 4.32 times. This implies that, on average across the five case studies, the genetic algorithm hybrid load forecasting algorithm reduced the errors by approximately four times when compared to the fuzzy logic load forecasting algorithm only.

The same inferences can be made for the total energy error. The error before the genetic algorithm was implemented was 7.67 %. This was reduced to 1.06 % after implementing the genetic algorithm. Thus an improvement factor of 18.95 times was observed.

Based on the results achieved the following observations were made regarding the load profiles, the load forecasting algorithm and the corresponding errors (performance criteria):

- Week and weekend load profiles were very similar for the Eastern Cape Province and the East Campus load.
- Barnato Hall and the single plug point had a more discernable difference between the week and weekend load profiles.
- Chamber of Mines had the most distinct difference between the week and weekend load profiles.
- The algorithm was highly susceptible to fluctuations in the load profile.
- Both defined errors were increased when significant differences were observed between the test week and algorithm input week data.
- The peak error was influenced more by fluctuations in the power usage data than the total energy error.
- Due to the definition of the performance criteria, a surplus or deficit in peak forecast and total energy usage forecast could not be distinguished based on the numerical value only.
- The computation time for the load forecasting algorithm was, on average, less than two seconds. However, the genetic algorithm computation time was approximately two hours when implemented using a 2.00 GHz CPU with 2 GB RAM. This illustrated the need to operate the load forecasting algorithm in 'real-time', while the genetic algorithm needs to be implemented separately before integrating the results into the system.

Despite the errors varying substantially, the load forecasting algorithm was found to function satisfactorily. The genetic algorithm took an increased calculation time when compared to the actual load forecasting calculation time due to the number of variables, population size and the maximum number of generations. However, due to the improvement exhibited on the performance criteria, it is still recommended for use in the overall algorithm.

Chapter 5

Recommendations for Future Work

Chapter Overview: *Following the completion of testing of the algorithm, enhancements and recommendations were made to further the study. This chapter focuses on several enhancements to the load forecasting algorithm and recommendations that could improve the current performance of the algorithm. Where possible simple tests have been implemented to verify the recommendations.*

Nomenclature

The following applies to each of the equations in this Chapter.

$P_{forecast}$	is the forecast power usage,
$P_{measured}$	is the measured power usage,
n	is the maximum number of terms in the forecast period.

In order to enhance the current study several items could be investigated further to observe the effects on the results. These could include:

- Developing a separate model for public holidays instead of relying on the assumption that public holidays can be modelled as weekend days.
- Adding additional fuzzy inputs for weather phenomenon.
- Altering the fuzzy logic system structure by varying the number of membership functions, rules and defuzzification technique.

- Using different genetic algorithm operators, such as the fitness function, scaling and selection, as well as others.
- Implementation of a shorter sample period for the forecasts (and measurements for validation) to allow for a faster response to the undesired fluctuations on the load profile.

During the course of the study, several implementation issues gave rise to the following recommendations:

- Absolute errors should not be used for the existing error calculations such that errors can be determined to be either surplus of deficit of what is required.
- In order to accurately compare this method against those already implemented, an additional calculation needs to be performed, the mean absolute percentage error (MAPE).
- The inputs to the fuzzy logic systems can be altered such that the instantaneous error is considered, and can aid in improving the existing performance criteria.

5.1 Definition of New Performance Criteria

Re-evaluating the original performance criteria, the following indicates the requirements for the new performance criteria:

1. The difference between the forecast peak load and the measured peak load for a 24 hour period,

$$E_{peak} = \frac{\max(P_{forecast}) - \max(P_{measured})}{\max(P_{measured})} \times 100 \tag{5.1}$$

and

2. The difference in the total energy required for the forecast load and the measured load in a 24 hour period,

$$E_{total} = \int_1^n \frac{P_{forecast}(t) - P_{measured}(t)}{P_{measured}(t)} dt \times 100 \tag{5.2}$$

A sample of the new results is shown in Table 5.1, using the same input data as for Case 4 in Chapter 4. The results clearly indicate whether the forecast was greater or less than the measured power usage. This aids the user to easily interpret the results such that the correct procedure for decreasing or maintaining the load can be implemented.

Table 5.1: Load forecasting results using the new performance criteria for the load forecasting algorithm, before and after the genetic algorithm (GA) was implemented.

	Before GA		**After GA**	
	E_{peak} (%)	E_{total} (%)	E_{peak} (%)	E_{total} (%)
Monday	0.36	0.42	-0.18	-0.10
Tuesday	0.36	-0.53	-0.18	-0.25
Wednesday	0.36	-0.00	-0.18	-0.02
Thursday	0.36	1.26	-0.18	0.03
Friday	5.00	3.34	0.20	0.95
Week Average	*1.29*	*0.90*	*-0.11*	*0.12*
Saturday	2.00	-0.12	-1.00	-0.03
Sunday	2.00	-0.98	-1.00	0.00
Weekend Average	*2.00*	*-0.55*	*-1.00*	*-0.02*
AVERAGE	**1.49**	**0.48**	**-0.36**	**0.08**

When the performance criteria are negative, it implies that the forecast is below the measured load usage. When the performance criteria are positive, it implies that the forecast is above the measured load usage. The overall average for this test case is now -0.25 % as opposed to 0.37 % as it was before changing the performance criteria.

5.2 Using the Mean Absolute Percentage Error

The MAPE, shown in Equation 2.2, is the most commonly used means of evaluating load forecasting algorithm performance, based on observations made in Chapter 2 and Appendix A.

This calculation provides the average of the absolute errors that has been calculated

between each of the input and output data points of the algorithm. The MAPE was calculated for each of the case studies in Chapter 4, after the genetic algorithm was implemented and is shown in Table 5.2.

Table 5.2: Mean absolute percentage error calculations for each of the presented case studies.

	Week average	**Weekend average**	**Average**
Case 1	25.72	29.46	26.79
Case 2	11.30	12.51	11.65
Case 3	19.69	19.70	19.69
Case 4	12.64	31.42	18.00
Case 5	15.71	18.19	16.42
AVERAGE	**17.01**	**22.06**	**18.51**

These results indicate that the load forecasting algorithm does not compare to the previously implemented algorithms. However, due to the defined performance criteria, it does not detract from the performance of the algorithm. The overall average MAPE was 18.15 % which is significant in comparison to the average MAPE of the fuzzy logic systems described in Chapter 2 and Appendix A. In future studies other methods need to be employed to reduce the MAPE as well as maintain the existing performance criteria.

5.3 Revising the Fuzzy Logic Systems

The fuzzy logic systems used in the load forecasting algorithm can be simplified and enhanced by altering the input sets as well as incorporating additional assumptions. If it is assumed that the historic power input is always a full 24 hour period (meaning 48 data points at half hour intervals) starting from 12:00 then the 'Time' input fuzzy set can be removed.

Using Equation 5.3 the instantaneous error (E_{inst}) can be determined. The instantaneous error can be incorporated as an input to the fuzzy logic systems such that when the error is large, the forecast power usage is adjusted accordingly to ensure a more accurate prediction. The revise load forecasting algorithm and associated features could be structured as shown in Figure 5.1.

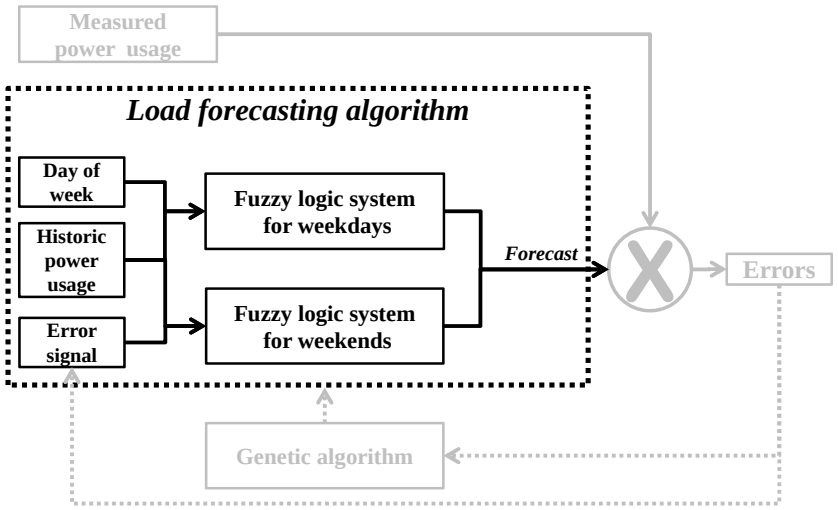


Figure 5.1: Structure of the revised load forecasting algorithm (in black) and the accompanying performance evaluation and optimisation processes (in grey).

$$E_{inst} = P_{forecast} - P_{measured} \tag{5.3}$$

A flow diagram to show the proposed logic flow for the algorithm is shown in Figure 5.2, with the output shown in blue. The error signal is initially assumed to be zero, however after the first week this will be replaced by the calculated values. The principle of the algorithm is the same as the original algorithm; however the error signal from the previous week will be fed forward to the current week to aid with the forecasting.

It is predicted that advantages to implementing this recommendation include:

- Fewer variable parameters, thus faster genetic algorithm computation time,
- Uses the defined performance criteria, and
- Reduced MAPE.

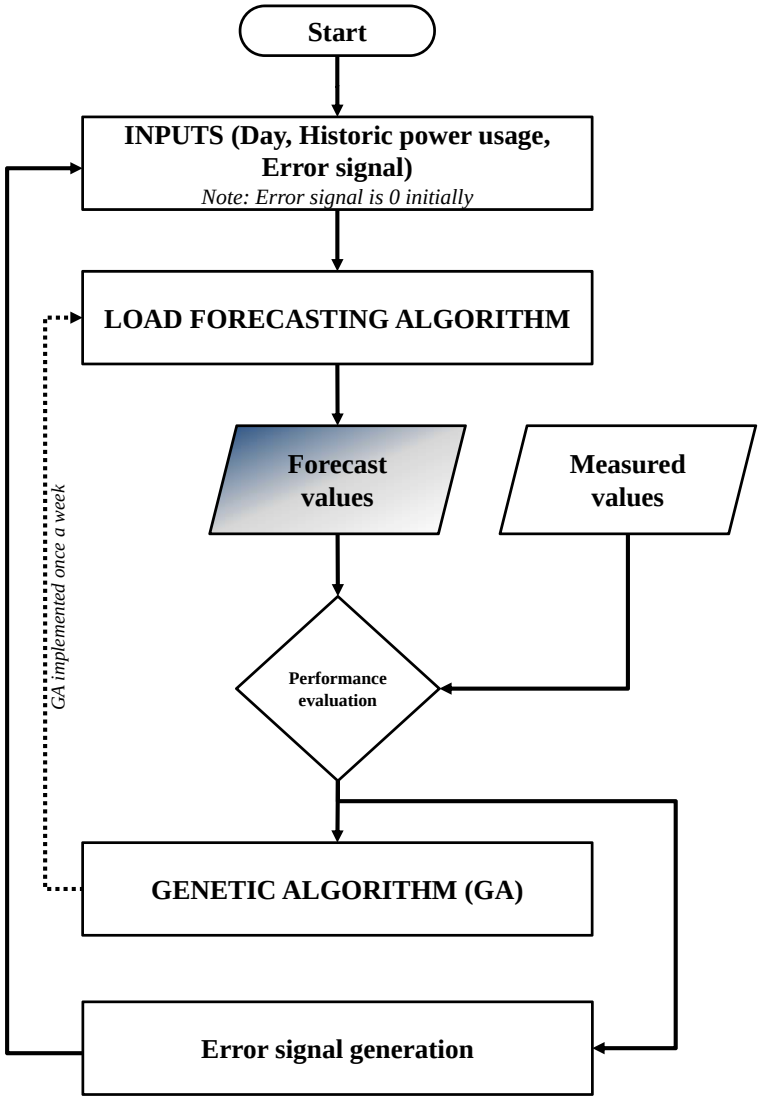


Figure 5.2: Flow diagram showing the proposed logic for the revised load forecasting algorithm.

The greatest disadvantage of implementing this recommendation is the necessity for more input data. It requires recall of the results from the previous week to implement the current week. A sample solution using this proposed method is shown in Table 5.3. The revised fuzzy logic systems in this test case had 33 parameters as opposed to the 44 in the original algorithm.

Table 5.3: Load forecasting results using the revised fuzzy logic systems (FLS) and the old method for comparison.

	Original FLS		**Revised FLS**	
	E_{peak} (%)	E_{total} (%)	E_{peak} (%)	E_{total} (%)
Monday	0.33	0.09	0.33	0.01
Tuesday	3.58	1.25	0.33	0.56
Wednesday	5.18	0.17	8.14	1.70
Thursday	5.18	1.65	8.14	0.03
Friday	0.33	0.68	0.88	0.97
Week Average	*2.92*	*0.77*	*3.56*	*0.65*
Saturday	0.75	0.09	0.75	0.18
Sunday	0.75	0.07	0.75	0.40
Weekend Average	*0.75*	*0.08*	*0.75*	*0.29*
AVERAGE	**2.30**	**0.57**	**2.76**	**0.55**

The original load forecasting algorithm yielded an overall average error (between E_{peak} and E_{total}) of 1.13 % while the revised algorithm yielded an overall error of 1.31 %. The revised algorithm computation time was approximately 30 minutes less than that for the original algorithm. This indicates that the revised method shows promise and could yield better results if taken further. The revised algorithm is also favourable when the MAPE is compared between the original algorithm and the revised algorithm. The original algorithm had an average MAPE of 50.65 % while the revised algorithm had an average MAPE of 29.20 %. This is an improvement by a factor of 1.7 times.

Chapter 6

Conclusion

The current energy short-fall that is being experienced world-wide, especially in South Africa, could potentially be improved by the concept of a smart grid. This is partially achieved by using energy management schemes such that the total and peak energy usage is monitored and maintained at a desired level. This can be indirectly implemented by load forecasting. Load forecasting would raise the user awareness to power usage, and also afford the power producer the knowledge of the total and peak energy required.

Load forecasting has been implemented many times in the past, using a variety of techniques. Short to medium term forecasters (an hour-ahead up to a month-ahead forecast) were considered and compared. It was discovered that the classic statistical methods yielded the most accurate forecasts, however required the greatest quantity of input data. It was also observed that combining different methods (called hybrid systems) yielded more accurate results. This led to the dissertation hypothesis - *fuzzy logic systems for load forecasting can be improved by incorporating an optimisation technique.* For this research, the optimisation technique used was genetic algorithms.

A load forecasting algorithm was developed, fulfilling certain assumptions and constraints, in MATLAB® using the fuzzy logic and optimisation toolboxes to generate the desired subsystems to make up the algorithm. The inputs to the algorithm were the day (week or weekend), time of day and the historical power usage. The performance criteria for the algorithm were defined to be the peak energy error and the total energy error, both in a 24 hour period.

During testing, data from one week prior to the test as the input was found to yield the most accurate results, by a factor of 1.4 times when compared to the other test cases. Using this knowledge, the algorithm was tested on five different test cases. They were the Eastern Cape Province in South Africa, East campus at the University, Barnato Hall student residence at the University and Chamber of Mines engineering building at the University, as well as a single plug point with a variable load. The algorithm indicated favourable results using the *a priori* knowledge for configuration. However, the genetic algorithm indicated a marked improvement to the results.

The most drastic improvement was observed for the Eastern Cape Province case, showing an improvement of 8.44 and 57.75 times for the peak energy error and total energy error respectively. The lowest improvement factors were for the Chamber of Mines case, still showing an improvement of 3.55 and 4.75 times for each of the above errors respectively. This was due to the load forecasting algorithm being designed for the Chamber of Mines case data originally.

The average peak energy error for all the presented cases, before implementing the genetic algorithm, was 7.22 %. This was reduced to 2.10 %, thus showing an improvement of 4.32 times. The average total energy error, before implementing the genetic algorithm, was 7.67 %. Implementing the genetic algorithm reduced this error to 1.06 % indicating an improvement of 18.95 times. Thus the hypothesis was proven to be correct. Due to the computation time of the genetic algorithm (approximately two hours), the optimisation loop could not be operated in real-time. The fuzzy logic load forecaster computed the forecast in approximately two seconds and thus could be operated in real-time if required.

Enhancements and recommendations were proposed in order to further the research and potentially improve the performance of the load forecasting algorithm. Where possible, basic tests were performed to substantiate the recommendations.

Appendix A

Detailed Comparison of Load Forecasting Implementations

Chapter Overview: *Several different methods of load forecasting have been implemented in the past. This chapter summarises and provides a comparison between different implementations for each of the load forecasting methods. The year the implementation was created; requirements for the method, forecast period and the mean absolute percentage error (MAPE) were some of the conditions for comparison. At least two different implementations are presented for each of the load forecasting methods to provide a more comprehensive comparison.*

Table A.1: Comparison of different implementation methods for load forecasting.

Author	**Year**	**Method Requirements**	**Further Information**	**Forecast period**	**MAPE**
		Statistical Models			
Asbury [39]	1975	Max. temperature for 3 days prior and historic power data	Regression analysis	1 month ahead	0.95 %
Moghram, Rahman [15]	1989	Power, temperature, dew point and wind speed	Regression analysis	24 hours ahead	2.78 %
Moghram, Rahman [15]	1989	4 weeks hourly power data, temperature	Time series analysis	24 hours ahead	0.53 %
Haida, Muto [16]	1994	Min. & max. temperature, humidity	Regression analysis	24 hours ahead	1.93 %
Liang, Cheng [29]	2000	Weekday power (for the system, at transformer and feeders) and temperature	Regression analysis	24 hours ahead	2.23 %
Amjady [40]	2001	Hourly power data and daily peak power data	Time series analysis	24 hours ahead	1.68 %

[... Continued on the next page ...]

Author	Year	Method Requirements	Further Information	Forecast period	MAPE
		Expert Systems			
Rahman, Bhatnagar [18]	1988	Hourly temperature and power data 5 weeks prior	-	24 hours ahead	3.29 %
Moghram, Rahman [15]	1989	Historic power for similarity comparison	-	24 hours ahead	1.22 %
Ho *et al.* [41]	1990	User input for selection of day, 5 year power database	-	24 hours ahead	1.64 %
		Artificial Neural Networks			
Park *et al.* [11]	1991	Temperature and power	Trained using Generalised delta rule	1 hour ahead	1.40 %
Kiartzis *et al.* [24]	1995	Power and min & max temperature (2 days prior)	Trained using Generalised delta rule	24 hours ahead	2.65 %
Kung *et al.* [25]	1998	Average, max and min temperatures, relative humidity and power (1 day prior)	Optimised using Genetic algorithms	24 hours ahead	3.83 %
Senjyu *et al.* [23]	2002	Power for 30 days prior and 60 days from the year before	Trained using Back propagation	1 hour ahead	1.18 %
[... Continued on the next page ...]					

Author	Year	Method Requirements	Further Information	Forecast period	MAPE
Liao [32]	2006	Historic power data, previous day max. & min. temperature, predicted temperature, rainfall	Optimised using Genetic Algorithms	24 hours ahead	1.52 %
Fuzzy Logic Systems					
Senjyu *et al.* [42]	1998	Power, temperature and precipitation (1 day prior)	-	24 hours ahead	2.53 %
Sachdeva, Verm [28]	2008	Min & max temp, precipitation, cloud, season, type of day and previous data	-	24 hours ahead	2.33 %
Combined Methods					
Liang, Cheng [29]	2000	Weekday power (for the system, at transformer and feeders) and temperature, previous day error calculations	Fuzzy-regression model	24 hours ahead	1.94 %
Srinivasan *et al.* [30]	1996	Type of day, date, daily max. & min. temperatures, past hourly load data	Fuzzy post-processor fuzzy-neural model	24 hours ahead	1.10 %
Liao [32]	2006	Historic power data, previous day max. & min. temperature, predicted temperature, rainfall	Parallel fuzzy-neural model	24 hours ahead	1.39 %

Appendix B

Important Definitions

Chapter Overview: *This chapter focuses on providing important definitions and principles to understand fuzzy logic systems and genetic algorithms in Chapter 3. The definitions for fuzzy logic systems were adapted from [22, 26, 27, 43, 44]. The definitions for genetic algorithms were adapted from [22, 34, 37, 45, 46] and the MATLAB® help files.*

B.1 Fuzzy Logic Systems

Universe of discourse: The universe of discourse is the range over which all the crisp values can lie. This is illustrated in Figure B.1. Typically the lower and upper bounds of the universe of discourse correspond with the minimum and maximum values of the system variables.

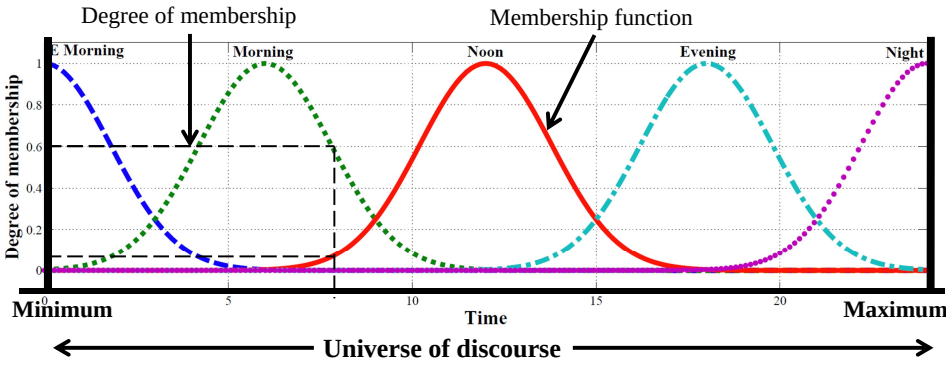


Figure B.1: Illustration of a universe of discourse, a fuzzy membership function and the degree of membership.

Crisp value: Crisp values are physical values that lie in the set of real numbers. These values most typically come from sensors and provide measured values of the variable.

Fuzzification: The fuzzification process (or fuzzifying) involves the mapping of the crisp values onto the corresponding universe of discourse. It then assigns the newly mapped values a degree of membership according to the membership functions within the fuzzy set.

Fuzzy value: A fuzzy value corresponds to the degree of membership assigned during the fuzzification process.

Membership function: The membership function defines a particular fuzzy set within the universe of discourse. The membership function assigns the degree of membership of the crisp values that fall into the set. This is illustrated in Figure B.1.

Degree of membership: Degree of membership is the indication of the sense of belonging to a particular linguistic variable. The higher the degree of membership the more the value belongs. An example of this is given in Figure B.1.

Fuzzy rule-base: A Mamdani-style fuzzy rule-base consists of a collection of IF-THEN antecedent-consequent style rules. The rule-base is constructed using *a priori* knowledge of the system so the behaviour can be linguistically described. These rules characterise the goals of the system.

Max-min inference: The max-min inference is a process whereby the fuzzy values and the fuzzy rule-base can be integrated in order to obtain output fuzzy values. This can be better described as follows:

Assuming A and B are two fuzzy sets within a universe of discourse U. They have membership functions μ_A and μ_B respectively. It should be noted that the union (fuzzy OR - Equation B.1) and intersection (fuzzy AND - Equation B.2) operations are important in the max-min inference.

$$\mu_{A \bigcup B}(u) = max\left\{\mu_A(u), \mu_B(u)\right\} \forall u \in U \tag{B.1}$$

$$\mu_{A \bigcap B}(u) = min\left\{\mu_A(u), \mu_B(u)\right\} \forall u \in U \tag{B.2}$$

Since the rules in the fuzzy rule-base have the form

$$\text{IF } a \text{ is } A \text{ AND } b \text{ is } B \text{ THEN } u = C$$

and each rule is combined by the fuzzy OR, then the rule can be mathematically inferred as:

$$\mu_C(u) = max\left[min\left\{\mu_A(u), \mu_B(u)\right\}\right] \tag{B.3}$$

Defuzzification: The defuzzification process (or defuzzifying) implies the output fuzzy values are reverse mapped into the real set, and thus a crisp value is obtained. An example of defuzzification can be seen in Figure 3.6, from Chapter 3.

B.2 Genetic Algorithms

Chromosome: A chromosome is the set of input variables for the genetic algorithm. This can be considered the genetic material constituting the solution you are solving for.

Population: The genetic population is a randomly generated group of chromosomes

Fitness function: The fitness function is a performance index such that the fittest chromosomes in the population will survive to the next generation. The fitter the parent, the greater the probability of selection for reproduction.

Fitness scaling: Fitness scaling converts the scores achieved from the fitness function to a range suitable for the function selected for reproduction.

Rank scaling: Rank scaling orders each individual according to their fitness, so the strongest come first and the weakest last.

Fitness function tolerance: The fitness function tolerance defines the stopping condition for the genetic algorithm. When the change in the weighted fitness function value becomes less than the fitness function tolerance the genetic algorithm stops.

Reproduction: Reproduction is the process where a selected chromosome is combined with another chromosome (the parents) to form a new chromosome of different genetic composition (the offspring).

Stochastic uniform: The stochastic uniform function implies that a good mix of genes (weak and strong) would be selected at a uniform random interval for breeding.

Parents: The parents are the selected chromosomes for reproduction.

Offspring: The offspring are the results of reproduction between two parents (crossover), or a single parent (mutation).

Crossover: Crossover determines the amount of genetic material from each parent that contributes to the new offspring. This is illustrated in Figure B.2.

Figure B.2: Illustration of parental crossover in genetic reproduction.

Scattered function: The scattered function ensures that each of the offspring inherited a random amount of the two parents' genes thus giving rise to a diverse population.

Mutation: Mutation is the random change in a gene within the chromosome of the offspring, as illustrated in Figure B.3. This is controlled by the mutation rate. Mutation allows for greater survey of the problem space.

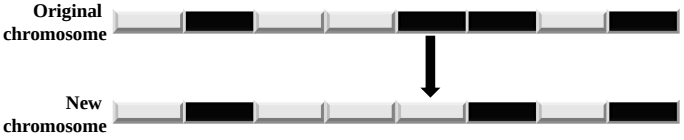


Figure B.3: Illustration of mutation of a chromosome on the fifth gene.

Adaptive scattered function: The adaptive scattered function allowed for a randomly generated number to be added to a random chromosome to make it differ from the original chromosome.

Generations: Once the new population has be created, the genetic algorithm progresses to the next generation. This is a factor that indicates how many times the population has changed.

Appendix C

MATLAB® Code Listings

Chapter Overview: *MATLAB® was used to implement the developed components of the load forecasting algorithm. This chapter contains the code listing for the load forecasting algorithm. The code shown is for the fuzzy logic system and genetic algorithm creation, definition of the fitness functions, initialisation of the algorithm as well as the calculation and display of the performance criteria errors.*

C.1 Fuzzy Logic System Creation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: createFLS.m
% Author:   Craig Carlson
% Description: Function to create the fuzzy logic systems
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ output ] = createFLS(input, choice)

t = input(1:10);    p = input(11:16);   pr = input(17:24);
w =  input(25:44);

forecast = newfis('Forecaster');

forecast = addvar(forecast,'input',   'Time',           [0      24]);
forecast = addmf (forecast,'input', 1,'EMorn','gaussmf',[t(1) t(6)]);
forecast = addmf (forecast,'input', 1,'Morn','gaussmf', [t(2) t(7)]);
forecast = addmf (forecast,'input', 1,'Noon','gaussmf', [t(3) t(8)]);
forecast = addmf (forecast,'input', 1,'Even','gaussmf', [t(4) t(9)]);
forecast = addmf (forecast,'input', 1,'Night','gaussmf',[t(5) t(10)]);

forecast = addvar(forecast,'input',   'Power',          [0      12]);
forecast = addmf (forecast,'input', 2,'Low','gaussmf',  [p(1) p(4)]);
forecast = addmf (forecast,'input', 2,'Med','gaussmf',  [p(2) p(5)]);
forecast = addmf (forecast,'input', 2,'High','gaussmf', [p(3) p(6)]);

forecast = addvar(forecast,'output',  'Prediction',    [0       12]);
forecast = addmf (forecast,'output',1,'V_Low','gaussmf',[pr(1) ...
    pr(5)]);
forecast = addmf (forecast,'output',1,'Low','gaussmf', [pr(2) pr(6)]);
forecast = addmf (forecast,'output',1,'Med','gaussmf', [pr(3) pr(7)]);
forecast = addmf (forecast,'output',1,'High','gaussmf',[pr(4) pr(8)]);

RULES = [ ...
     1 1 1 w(1)  1
     2 1 1 w(2)  1
     3 1 1 w(3)  1
     4 1 1 w(4)  1
     5 1 1 w(5)  1
     1 2 2 w(6)  1
     2 2 2 w(7)  1
```

```
40      3 2 2 w(8)  1
41      4 2 2 w(9)  1
42      5 2 2 w(10) 1
43      1 2 3 w(11) 1
44      2 2 3 w(12) 1
45      3 2 3 w(13) 1
46      4 2 3 w(14) 1
47      5 2 3 w(15) 1
48      1 3 4 w(16) 1
49      2 3 4 w(17) 1
50      3 3 4 w(18) 1
51      4 3 4 w(19) 1
52      5 3 4 w(20) 1 ];
53
54  forecast = addrule(forecast,RULES);
55  forecast.defuzzMethod = ('mom');
56
57  if choice == 1
58      writefis(forecast,'WeekForecaster');    output = 1;
59  elseif choice == 2
60      writefis(forecast,'WkndForecaster');    output = 1;
61  else
62      output = 0;
63  end
64
65  end
```

C.2 Genetic Algorithm Code

C.2.1 Genetic Algorithm Creation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: GA_Tool.m
% Author:   Craig Carlson
% Description: Function to generate the genetic algorithm parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,fval,exitflag,output,population,score] =
          GA_Tool(nvars,lb,ub,PopulationSize_Data,EliteCount_Data,...
          Generations_Data,InitialPopulation_Data,TolFun_Data,choice)

% Start with the default options
options = gaoptimset;
% Modify options setting
options = gaoptimset(options,'PopulationSize', PopulationSize_Data);
options = gaoptimset(options,'EliteCount', EliteCount_Data);
options = gaoptimset(options,'Generations', Generations_Data);
options = gaoptimset(options,'InitialPopulation', ...
    InitialPopulation_Data);
options = gaoptimset(options,'TolFun', TolFun_Data);
options = gaoptimset(options,'Display', 'final');

if choice == 1
    [x,fval,exitflag,output,population,score] = ...
                    ga(@FLSweek_GA,nvars,[],[],[],[],lb,ub,[],options);
elseif choice == 2
    [x,fval,exitflag,output,population,score] = ...
                    ga(@FLSwknd_GA,nvars,[],[],[],[],lb,ub,[],options);
else
    fprintf('Incorrect output selected!')
end
```

C.2.2 Fitness Functions

Weekday Fitness Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: FLSweek_GA.m
% Author:   Craig Carlson
% Description: Fitness function for the week day genetic algorithm.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function output = FLSweek_GA( input )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% global input_power output_power time
global TestMon TestTues TestWed TestThurs TestFri time
global OutMon  OutTues  OutWed  OutThurs  OutFri

createFLS(input,1);
a = readfis('WeekForecaster');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Prediction section
for i = 1:1:48
    t = (i-1)/2;
    % Perform predictions
    predMon(i)   = evalfis([t,TestMon(i)],   a);
    predTues(i)  = evalfis([t,TestTues(i)],  a);
    predWed(i)   = evalfis([t,TestWed(i)],   a);
    predThurs(i) = evalfis([t,TestThurs(i)], a);
    predFri(i)   = evalfis([t,TestFri(i)],   a);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculation Section
% Peak Energy Error
pkErrMon   = (abs(max(predMon)   - max(OutMon))   /max(OutMon))   *100;
pkErrTues  = (abs(max(predTues)  - max(OutTues))  /max(OutTues))  *100;
pkErrWed   = (abs(max(predWed)   - max(OutWed))   /max(OutWed))   *100;
pkErrThurs = (abs(max(predThurs) - max(OutThurs)) /max(OutThurs))*100;
pkErrFri   = (abs(max(predFri)   - max(OutFri))   /max(OutFri))   *100;

% Total Energy Error
% Calculate total energy required by output
```

```
39 enMon    = trapz(time,OutMon);         enTues   = trapz(time,OutTues);
40 enWed    = trapz(time,OutWed);         enThurs  = trapz(time,OutThurs);
41 enFri    = trapz(time,OutFri);
42 % Calculate total energy required by prediction
43 enpMon    = trapz(time,predMon);       enpTues  = trapz(time,predTues);
44 enpWed    = trapz(time,predWed);       enpThurs = trapz(time,predThurs);
45 enpFri    = trapz(time,predFri);
46 enErrMon    = (abs(enpMon    - enMon)   / enMon)    *100;
47 enErrTues   = (abs(enpTues   - enTues)  / enTues)   *100;
48 enErrWed    = (abs(enpWed    - enWed)   / enWed)    *100;
49 enErrThurs = (abs(enpThurs - enThurs)/ enThurs) *100;
50 enErrFri    = (abs(enpFri    - enFri)   / enFri)    *100;
51
52 pkWeekAvg  = (pkErrMon+pkErrTues+pkErrWed+pkErrThurs+pkErrFri)/5;
53 enWeekAvg  = (enErrMon+enErrTues+enErrWed+enErrThurs+enErrFri)/5;
54 WeekAvg = (pkWeekAvg+enWeekAvg)/2;
55 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57 output = WeekAvg;
58 end
```

Weekend Fitness Function

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % Filename: FLSwknd_GA.m
3  % Author:   Craig Carlson
4  % Description: Fitness function for the weekend day genetic algorithm.
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7  function output = FLSwknd_GA( input )
8  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10 % global input_power output_power time
11 global TestSat TestSun time
12 global OutSat  OutSun
13
14 createFLS(input,2);
15 b = readfis('WkndForecaster');
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18 % Prediction section
```

```
for i = 1:1:48
    t = (i-1)/2;
    % Perform predictions
    predSat(i)   = evalfis([t,TestSat(i)],   b);
    predSun(i)   = evalfis([t,TestSun(i)],   b);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculation Section
% Peak Energy Error
pkErrSat   = (abs(max(predSat) - max(OutSat)) /max(OutSat)) *100;
pkErrSun   = (abs(max(predSun) - max(OutSun)) /max(OutSun)) *100;

% Total Energy Error
% Calculate total energy required by output
enSat   = trapz(time,OutSat);          enSun   = trapz(time,OutSun);
% Calculate total energy required by prediction
enpSat   = trapz(time,predSat);        enpSun   = trapz(time,predSun);
enErrSat   = (abs(enpSat   - enSat)  / enSat)   *100;
enErrSun   = (abs(enpSun   - enSun)  / enSun)   *100;

pkEndAvg   = (pkErrSat+pkErrSun)/2;
enEndAvg   = (enErrSat+enErrSun)/2;
WkndAvg    = (pkEndAvg+enEndAvg)/2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
output = WkndAvg;
end
```

C.3 Algorithm Initialisation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: InitialSettings.m
% Author:   Craig Carlson
% Description: Configuration for the initial settings for the fuzzy
% logic systems and genetic algorithm.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setting the lower and upper limits for the GA.
nRules = 20;                                        % Number of rules
tl  = [0.01 0.01 0.01 0.01 0.01 0 0 0 0 0];
pl = [0.01 0.01 0.01 0 0 0];
prl = [0.01 0.01 0.01 0.01 0 0 0 0];           wl = zeros(1,nRules);
tu  = [5 5 5 5 5 25 25 25 25 25];              pu = [4 4 4 13 13 13];
pru = [4 4 4 4 13 13 13 13];                   wu = ones(1,nRules);
% Configuring the necessary information for the GA
nvar = 44;           lower = [tl pl prl wl];   upper = [tu pu pru wu];
pop  = 120;          elite = 4;                     generations = 200;
ftol = 1e-9;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Creating the first pass at the desired input for weekday predictions
t  = [1.8 1.8 1.8 1.8 1.8 0 6 12 18 24];     p = [1.8 0.9 1.8 0 6 12];
pr = [3.6 0.9 0.9 3.6 0 5.5 6.5  12];        w = ones(1,nRules);
% Creating the first pass at the desired input for weekend predictions
t2  = [4.2 2.0 3.2 2.0 3.9 0 6 12 18 24];   p2 = [2.5 2.5 2.5 0 6 12];
pr2 = [3.8 1.8 1.8 3.8 0 5.5 6.5  12];      w2 = ones(1,nRules);

week_input = [t p pr w];                  wknd_input = [t2 p2 pr2 w2];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.4 Algorithm Calculations and Results

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: AlgResults.m
% Author:   Craig Carlson
% Description: Error calculation and display for algorithm testing.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input data
time = 0:0.5:23.5;
data = dlmread('Data.csv',',','B2..B1009');

variable = max(data)/12;
data = data/variable;                                   % Normalise data

for i = 1:1:48
         TestMon(i)    = data(7  *48+i);
         TestTues(i)   = data(8  *48+i);
         TestWed(i)    = data(9  *48+i);
         TestThurs(i)  = data(10 *48+i);
         TestFri(i)    = data(11 *48+i);
         TestSat(i)    = data(12 *48+i);
         TestSun(i)    = data(13 *48+i);

         OutMon(i)     = data(14 *48+i);
         OutTues(i)    = data(15 *48+i);
         OutWed(i)     = data(16 *48+i);
         OutThurs(i)   = data(17 *48+i);
         OutFri(i)     = data(18 *48+i);
         OutSat(i)     = data(19 *48+i);
         OutSun(i)     = data(20 *48+i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
createFLS(week_input,1);              a = readfis('WeekForecaster');
createFLS(wknd_input,2);              b = readfis('WkndForecaster');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:1:48
    t = (i-1)/2;

```

```
41      % Perform predictions
42      predMon(i)    = evalfis([t,TestMon(i)],    a);
43      predTues(i)   = evalfis([t,TestTues(i)],   a);
44      predWed(i)    = evalfis([t,TestWed(i)],    a);
45      predThurs(i)  = evalfis([t,TestThurs(i)],  a);
46      predFri(i)    = evalfis([t,TestFri(i)],    a);
47      predSat(i)    = evalfis([t,TestSat(i)],    b);
48      predSun(i)    = evalfis([t,TestSun(i)],    b);
49
50      % Part calculation for the MAPE
51      errMon(i)     =  OutMon(i)   - predMon(i);
52      errTues(i)    =  OutTues(i)  - predTues(i);
53      errWed(i)     =  OutWed(i)   - predWed(i);
54      errThurs(i)   =  OutThurs(i) - predThurs(i);
55      errFri(i)     =  OutFri(i)   - predFri(i);
56      errSat(i)     =  OutSat(i)   - predSat(i);
57      errSun(i)     =  OutSun(i)   - predSun(i);
58  end
59
60  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61  % Calculation Section
62  % Peak Energy Error
63  pkErrMon   = (abs(max(predMon)   - max(OutMon))   /max(OutMon))  *100;
64  pkErrTues  = (abs(max(predTues)  - max(OutTues))  /max(OutTues)) *100;
65  pkErrWed   = (abs(max(predWed)   - max(OutWed))   /max(OutWed))  *100;
66  pkErrThurs = (abs(max(predThurs) - max(OutThurs)) /max(OutThurs))*100;
67  pkErrFri   = (abs(max(predFri)   - max(OutFri))   /max(OutFri))  *100;
68  pkErrSat   = (abs(max(predSat)   - max(OutSat))   /max(OutSat))  *100;
69  pkErrSun   = (abs(max(predSun)   - max(OutSun))   /max(OutSun))  *100;
70
71  pkWeekAvg  = (pkErrMon+pkErrTues+pkErrWed+pkErrThurs+pkErrFri)/5;
72  pkEndAvg   = (pkErrSat+pkErrSun)/2;
73  pkAVG = ((pkWeekAvg*5)+(pkEndAvg*2))/7;
74
75  % Total Energy Error
76  % Calculate total energy required by output
77  enMon   = trapz(time,OutMon);       enTues  = trapz(time,OutTues);
78  enWed   = trapz(time,OutWed);       enThurs = trapz(time,OutThurs);
79  enFri   = trapz(time,OutFri);
80  enSat   = trapz(time,OutSat);       enSun   = trapz(time,OutSun);
81  % Calculate total energy required by prediction
82  enpMon   = trapz(time,predMon);    enpTues  = trapz(time,predTues);
83  enpWed   = trapz(time,predWed);    enpThurs = trapz(time,predThurs);
84  enpFri   = trapz(time,predFri);
```

```
85  enpSat   = trapz(time,predSat);    enpSun   = trapz(time,predSun);
86  enErrMon   = (abs(enpMon   - enMon)  / enMon)   *100;
87  enErrTues  = (abs(enpTues  - enTues) / enTues)  *100;
88  enErrWed   = (abs(enpWed   - enWed)  / enWed)   *100;
89  enErrThurs = (abs(enpThurs - enThurs)/ enThurs) *100;
90  enErrFri   = (abs(enpFri   - enFri)  / enFri)   *100;
91  enErrSat   = (abs(enpSat   - enSat)  / enSat)   *100;
92  enErrSun   = (abs(enpSun   - enSun)  / enSun)   *100;
93
94  enWeekAvg  = (enErrMon+enErrTues+enErrWed+enErrThurs+enErrFri)/5;
95  enEndAvg   = (enErrSat+enErrSun)/2;
96  enAVG = ((enWeekAvg*5)+(enEndAvg*2))/7;
97
98
99  WeekAvg = (pkWeekAvg+enWeekAvg)/2;
100 WkndAvg = (pkEndAvg+enEndAvg)/2;
101 AVG     = (WeekAvg+WkndAvg)/2;
102
103
104 % Mean Absolute Percentage Error
105 mapeMon   = mean(((sum(abs(errMon))   / sum(OutMon))   *100));
106 mapeTues  = mean(((sum(abs(errTues))  / sum(OutTues))  *100));
107 mapeWed   = mean(((sum(abs(errWed))   / sum(OutWed))   *100));
108 mapeThurs = mean(((sum(abs(errThurs)) / sum(OutThurs)) *100));
109 mapeFri   = mean(((sum(abs(errFri))   / sum(OutFri))   *100));
110 mapeSat   = mean(((sum(abs(errSat))   / sum(OutSat))   *100));
111 mapeSun   = mean(((sum(abs(errSun))   / sum(OutSun))   *100));
112
113
114 mapeWeekAvg = (mapeMon+mapeTues+mapeWed+mapeThurs+mapeFri)/5;
115 mapeWkndAvg = (mapeSat + mapeSun)/2;
116 mapeAVG     = (mapeWeekAvg*5 + mapeWkndAvg*2)/7;
117 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
118 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
119
120 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
121 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
122 % Printing out results to the command window
123 fprintf('*********************************************************\n')
124 fprintf('                      PkE            EnE                ...
        Average\n')
125 fprintf('*********************************************************\n')
126 fprintf('Monday             %6.2f          %6.2f\n', pkErrMon,    ...
        enErrMon)
```

```
127 fprintf('Tuesday            %6.2f          %6.2f\n', pkErrTues,  ...
        enErrTues)
128 fprintf('Wednesday          %6.2f          %6.2f\n', pkErrWed,   ...
        enErrWed)
129 fprintf('Thursday           %6.2f          %6.2f\n', pkErrThurs, ...
        enErrThurs)
130 fprintf('Friday             %6.2f          %6.2f\n', pkErrFri,   ...
        enErrFri)
131 fprintf('-------------------------------------------------------\n')
132 fprintf('WeekAvg            %6.2f          %6.2f              ...
        %6.2f\n',...
133                                                pkWeekAvg,  enWeekAvg, ...
                                                       WeekAvg)
134 fprintf('-------------------------------------------------------\n')
135 fprintf('Saturday           %6.2f          %6.2f\n', pkErrSat,   ...
        enErrSat)
136 fprintf('Sunday             %6.2f          %6.2f\n', pkErrSun,   ...
        enErrSun)
137 fprintf('-------------------------------------------------------\n')
138 fprintf('WkndAvg            %6.2f          %6.2f              ...
        %6.2f\n',...
139                                                  pkEndAvg,   enEndAvg, ...
                                                       WkndAvg)
140 fprintf('=======================================================\n')
141 fprintf('Average            %6.2f          %6.2f              ...
        %6.2f\n',...
142                                                             pkAVG,   ...
                                                                 enAVG, AVG)
143 fprintf('=======================================================\n')
144 fprintf('Weekday MAPE = %6.2f        and        Weekend MAPE = ...
        %6.2f\n',mapeWeekAvg, mapeWkndAvg)
145 fprintf('AVERAGE MAPE = %6.2f\n',    mapeAVG)
146 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
147 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix D

Additional Information for the Case Studies

Chapter Overview: *A week long view of the load profile and the results of the load forecasting algorithm give an indication of the performance of the algorithm. However to provide a clearer understanding of what is happening smaller intervals need to be given. This chapter presents the additional information required for the case studies in Chapter 4. This includes maps indicating the area covered for the tests, break-down of the load composition and the daily load profile graphs (Monday to Sunday). The area map for Case 1 was taken from Google Earth. The area maps for Case2, Case 3 and Case 4 were adapted from [47, 48] at the University of the Witwatersrand, Johannesburg South Africa (henceforth known as the University).*

D.1 Case 1 - Eastern Cape Province in South Africa

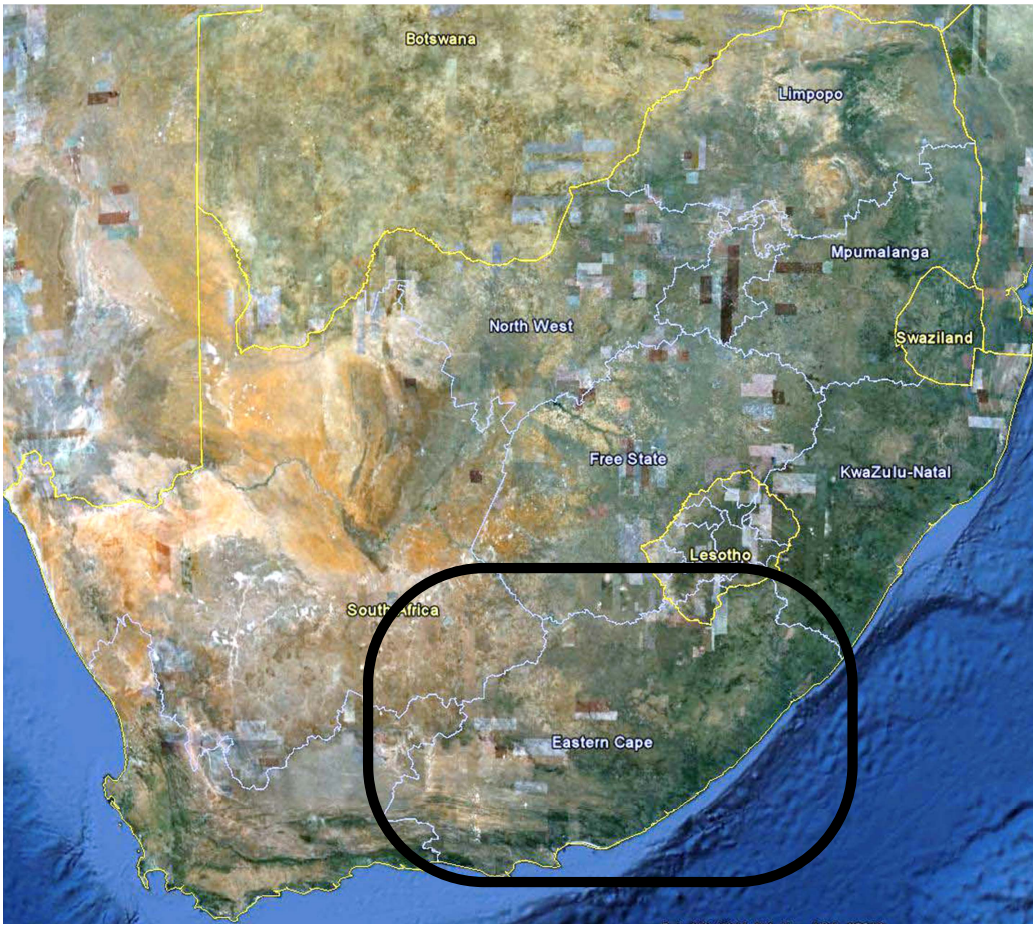


Figure D.1: Map of South Africa (taken from Google Earth) showing the area the load profile was taken from, for the Eastern Cape Province test.

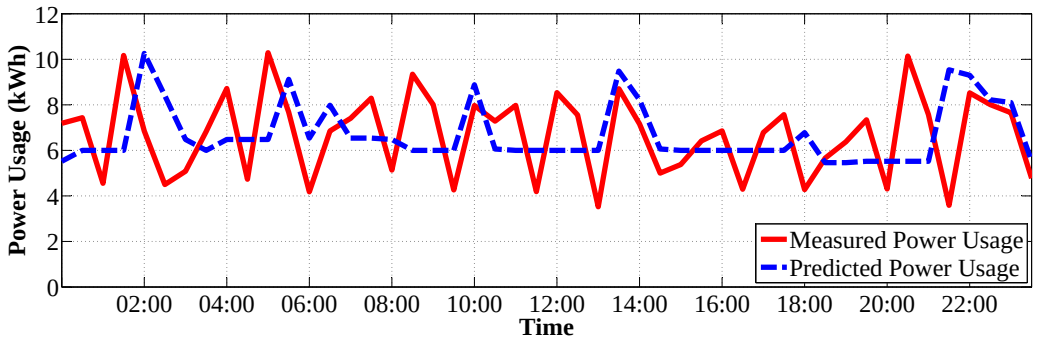


Figure D.2: Normalised load profile and the predicted power usage for the Eastern Cape Province for a general Monday.

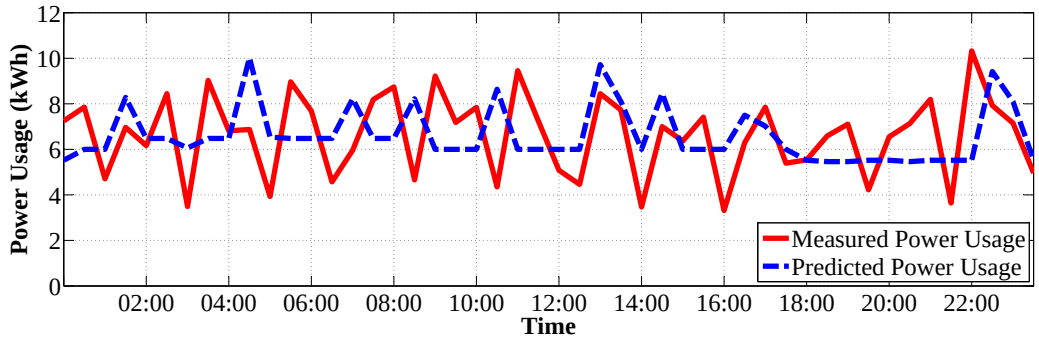


Figure D.3: Normalised load profile and the predicted power usage for the Eastern Cape Province for a general Tuesday.

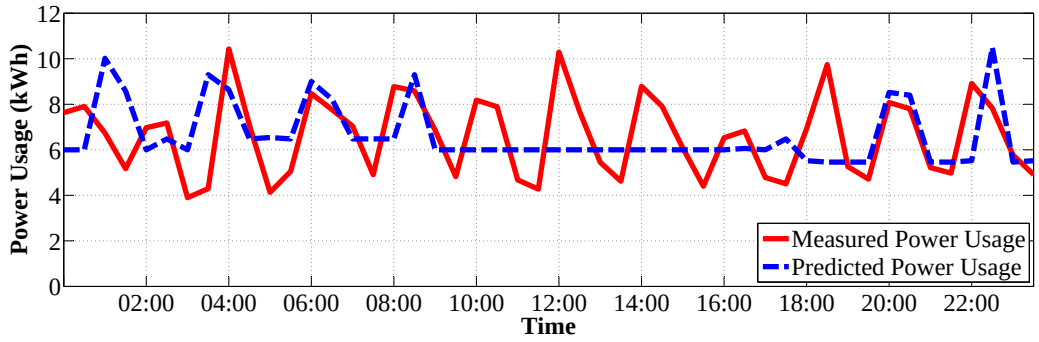


Figure D.4: Normalised load profile and the predicted power usage for the Eastern Cape Province for a general Wednesday.

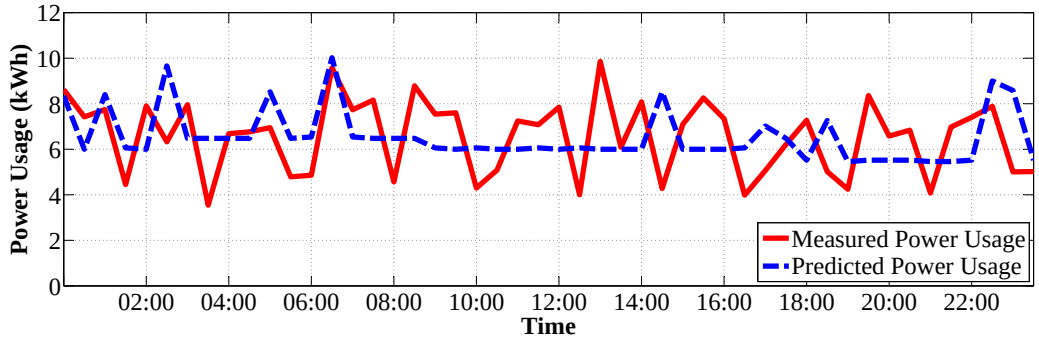


Figure D.5: Normalised load profile and the predicted power usage for the Eastern Cape Province for a general Thursday.

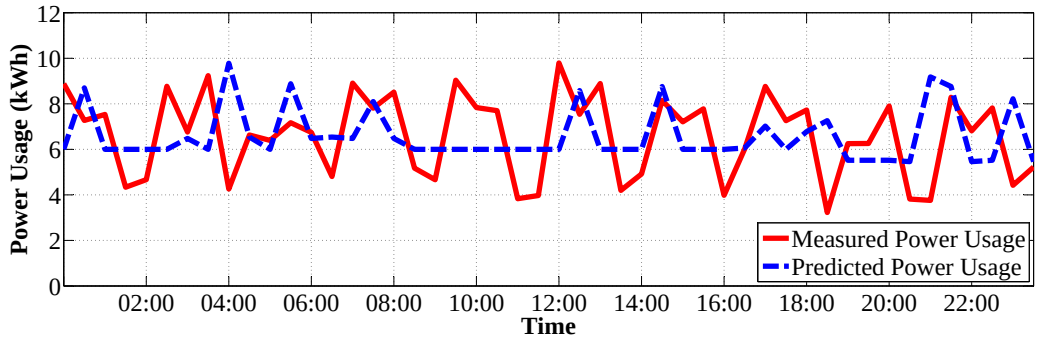


Figure D.6: Normalised load profile and the predicted power usage for the Eastern Cape Province for a general Friday.

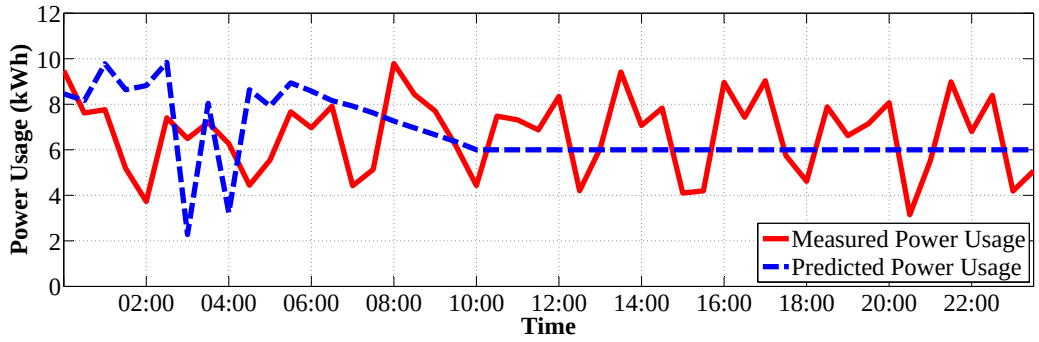


Figure D.7: Normalised load profile and the predicted power usage for the Eastern Cape Province for a general Saturday.

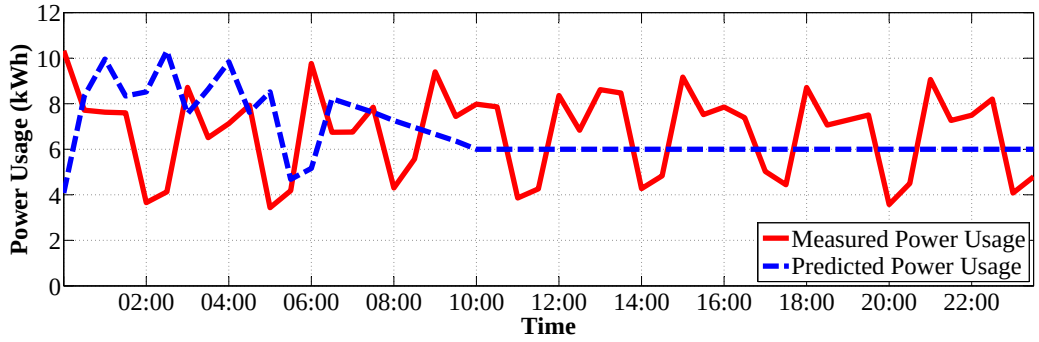


Figure D.8: Normalised load profile and the predicted power usage for the Eastern Cape Province for a general Sunday.

D.2 Case 2 - East Campus at the University

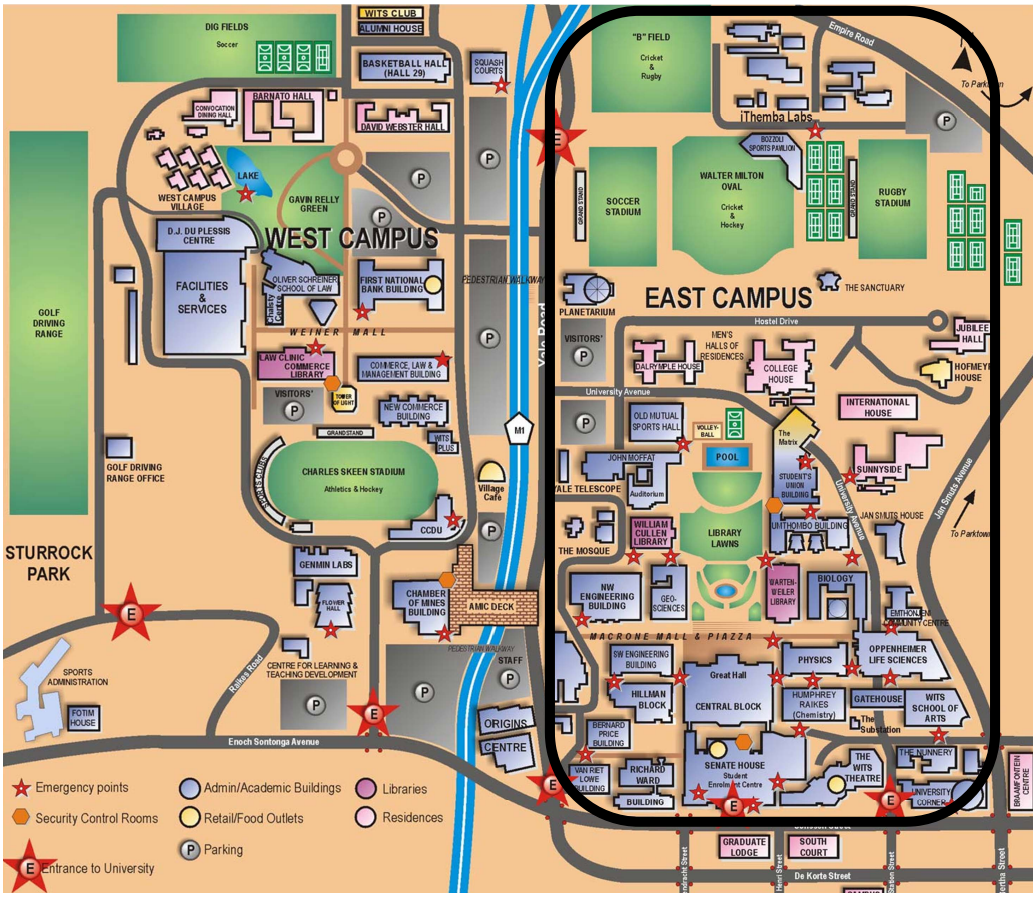


Figure D.9: Map of Main Campus at the University showing the area the load profile was taken from, for the East Campus test.

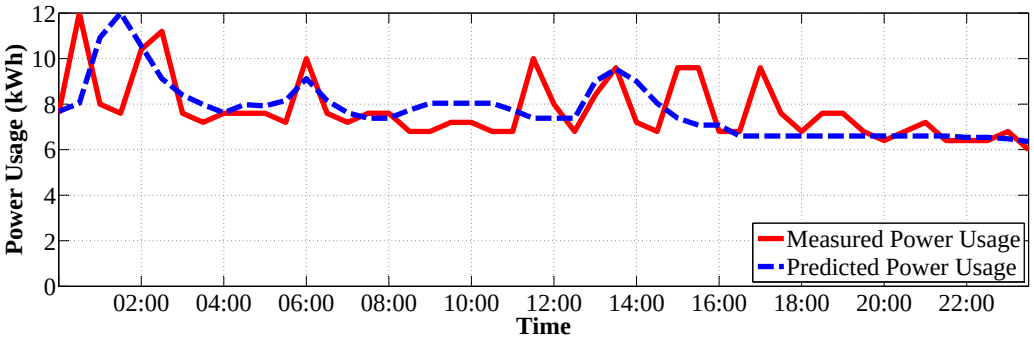


Figure D.10: Normalised load profile and the predicted power usage for East Campus at the University for a general Monday.

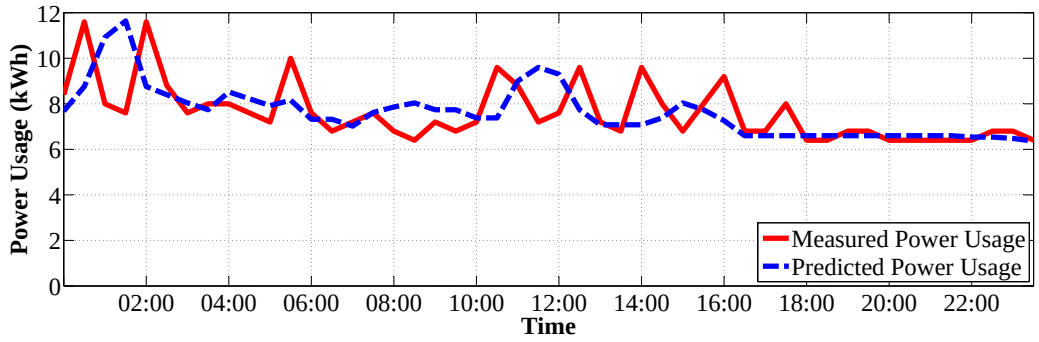


Figure D.11: Normalised load profile and the predicted power usage for East Campus at the University for a general Tuesday.

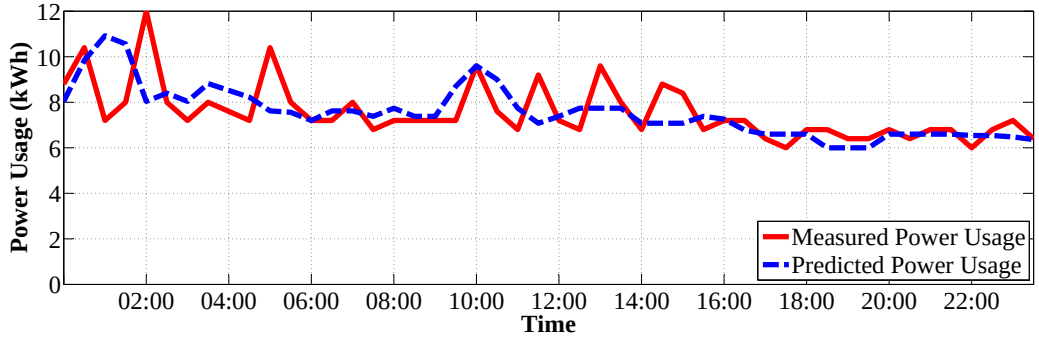


Figure D.12: Normalised load profile and the predicted power usage for East Campus at the University for a general Wednesday.

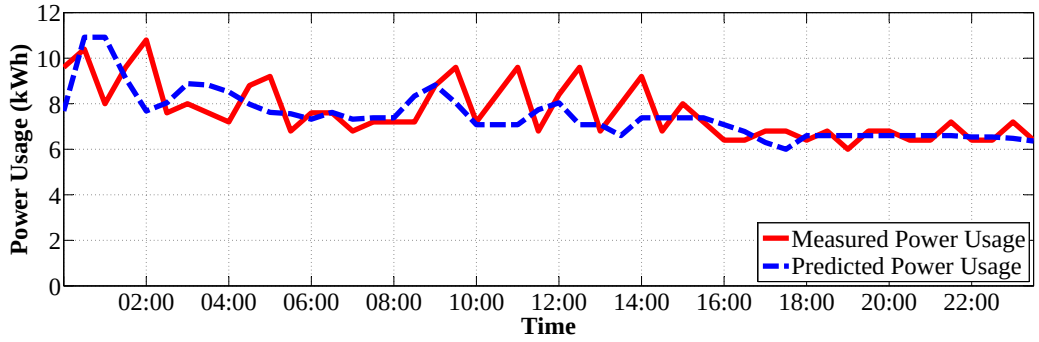


Figure D.13: Normalised load profile and the predicted power usage for East Campus at the University for a general Thursday.

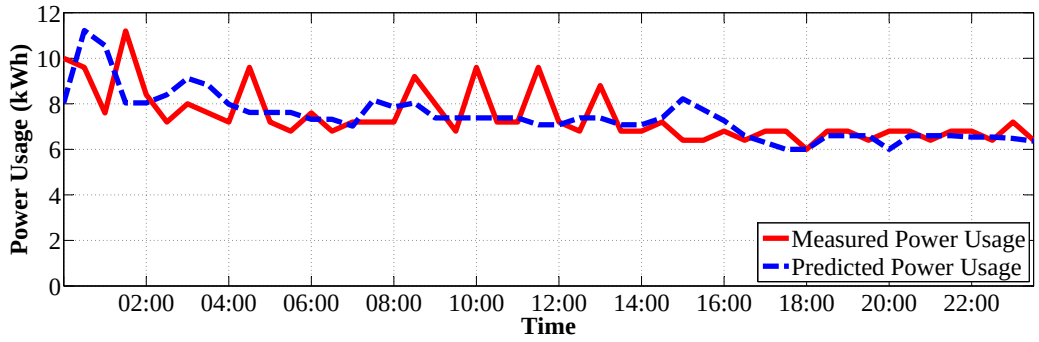


Figure D.14: Normalised load profile and the predicted power usage for East Campus at the University for a general Friday.

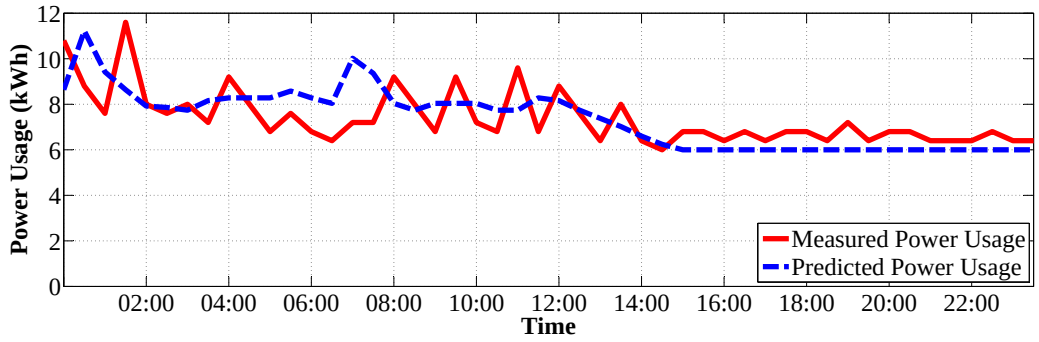


Figure D.15: Normalised load profile and the predicted power usage for East Campus at the University for a general Saturday.

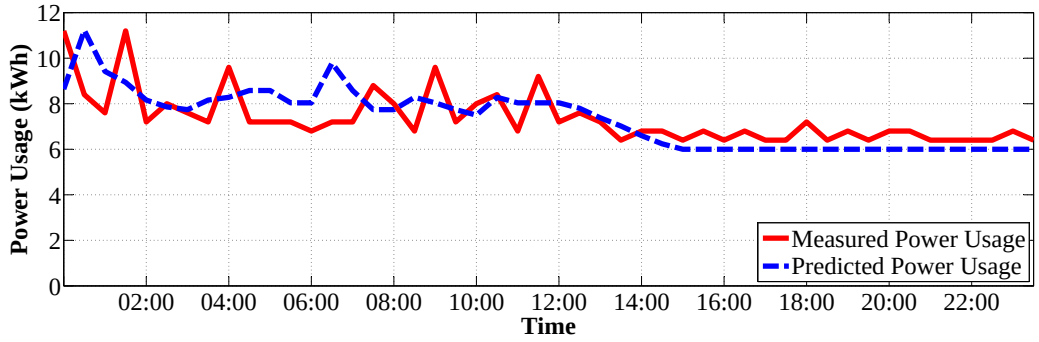


Figure D.16: Normalised load profile and the predicted power usage for East Campus at the University for a general Sunday.

D.3 Case 3 - Barnato Hall Student Residence at the University

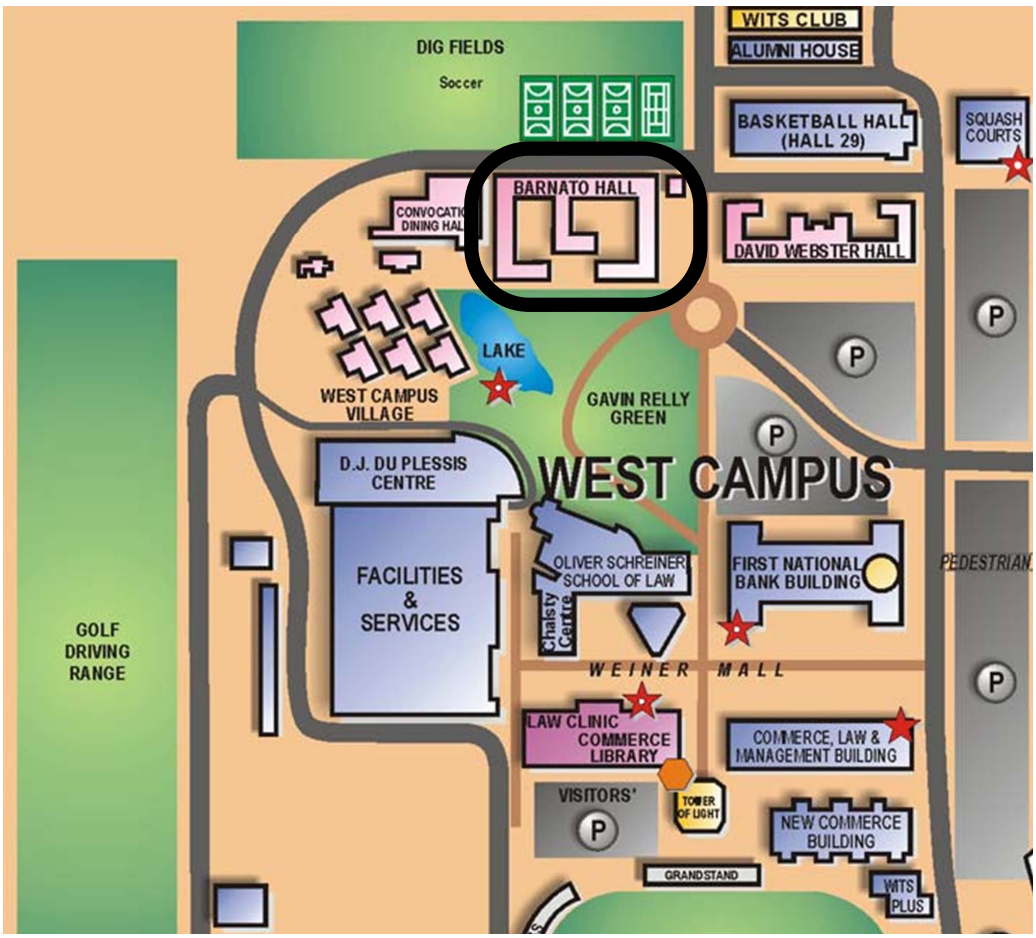


Figure D.17: Map of West Campus at the University showing the area the load profile was taken from, for the Barnato Hall student residence test.

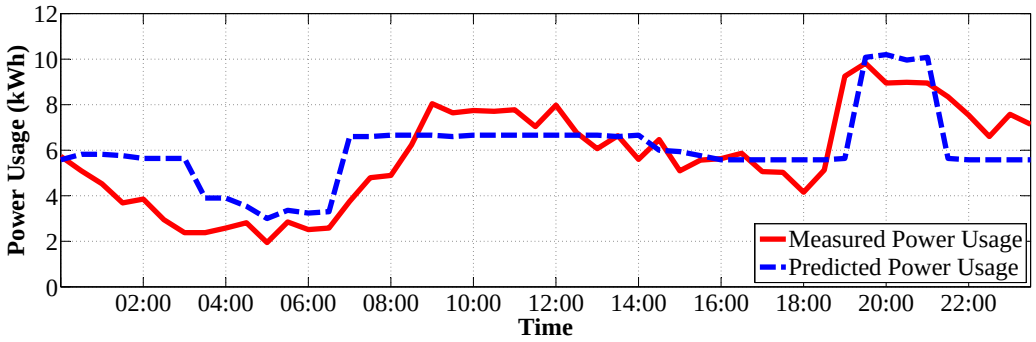


Figure D.18: Normalised load profile and the predicted power usage for the Barnato Hall student residence at the University for a general Monday.

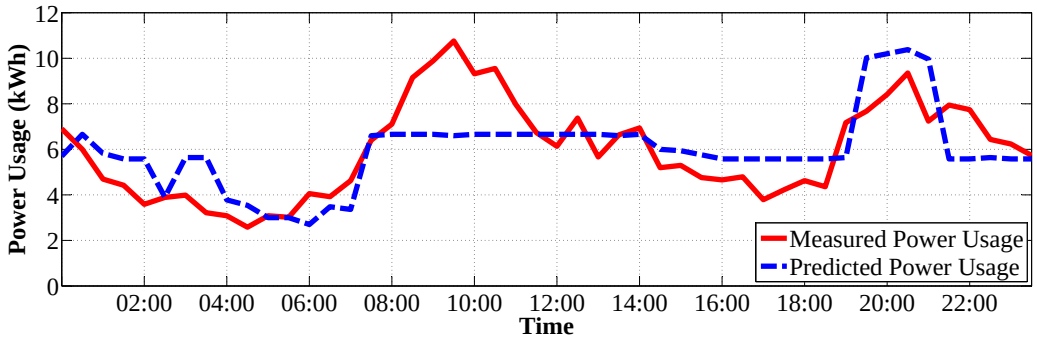


Figure D.19: Normalised load profile and the predicted power usage for the Barnato Hall student residence at the University for a general Tuesday.

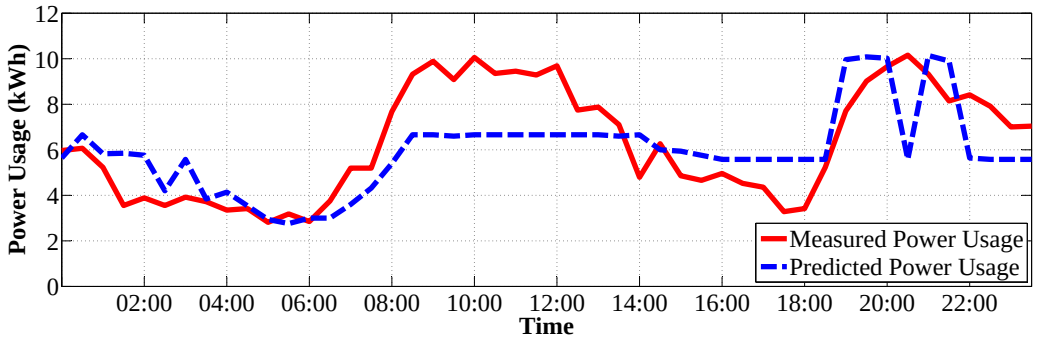


Figure D.20: Normalised load profile and the predicted power usage for the Barnato Hall student residence at the University for a general Wednesday.

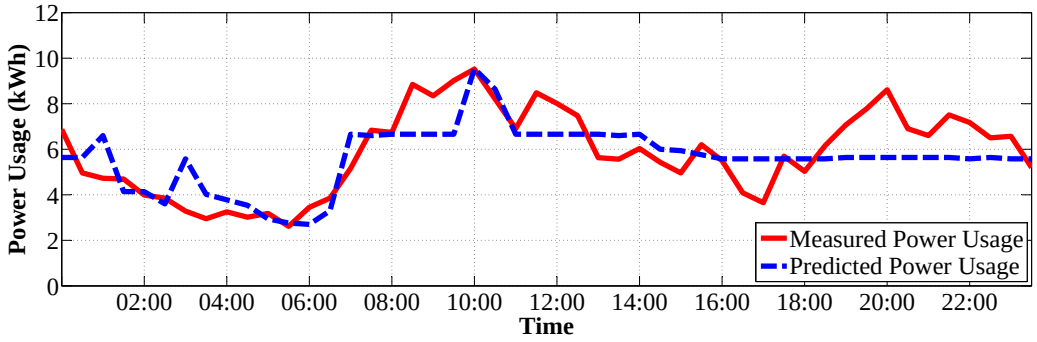


Figure D.21: Normalised load profile and the predicted power usage for the Barnato Hall student residence at the University for a general Thursday.

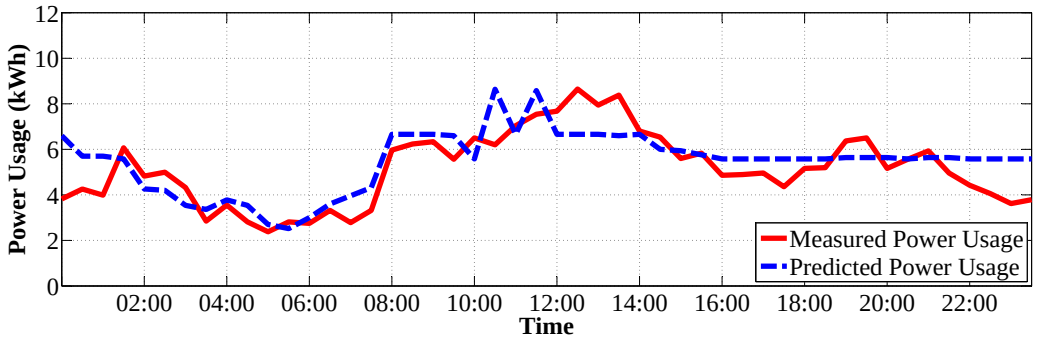


Figure D.22: Normalised load profile and the predicted power usage for the Barnato Hall student residence at the University for a general Friday.

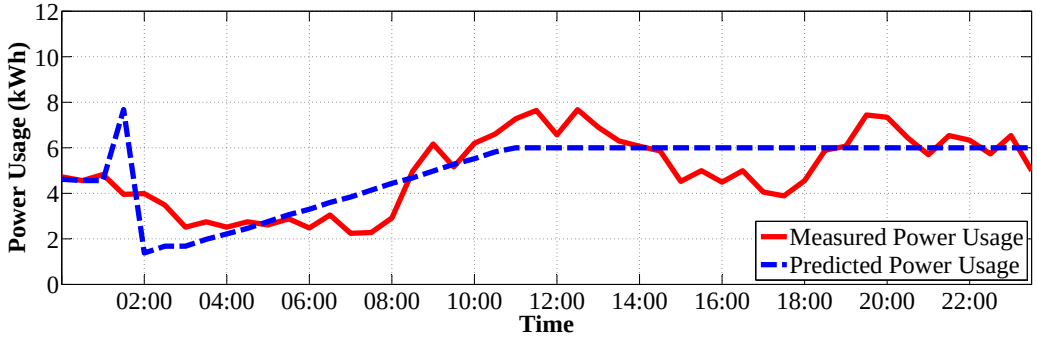


Figure D.23: Normalised load profile and the predicted power usage for the Barnato Hall student residence at the University for a general Saturday.

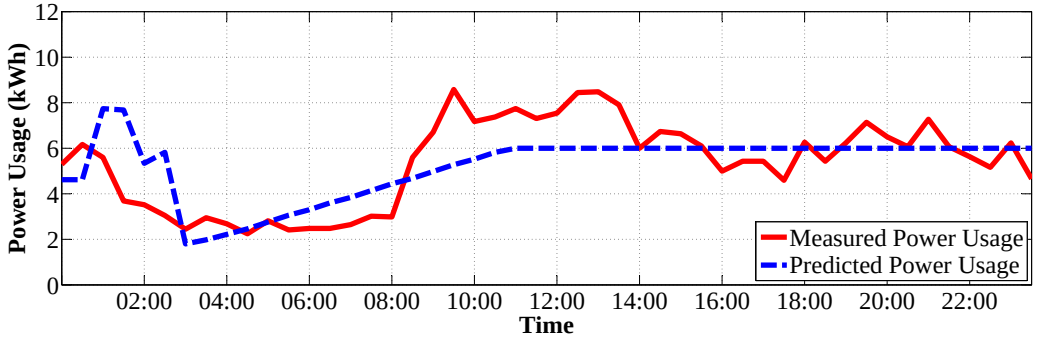


Figure D.24: Normalised load profile and the predicted power usage for the Barnato Hall student residence at the University for a general Sunday.

D.4 Case 4 - Chamber of Mines Building at the University

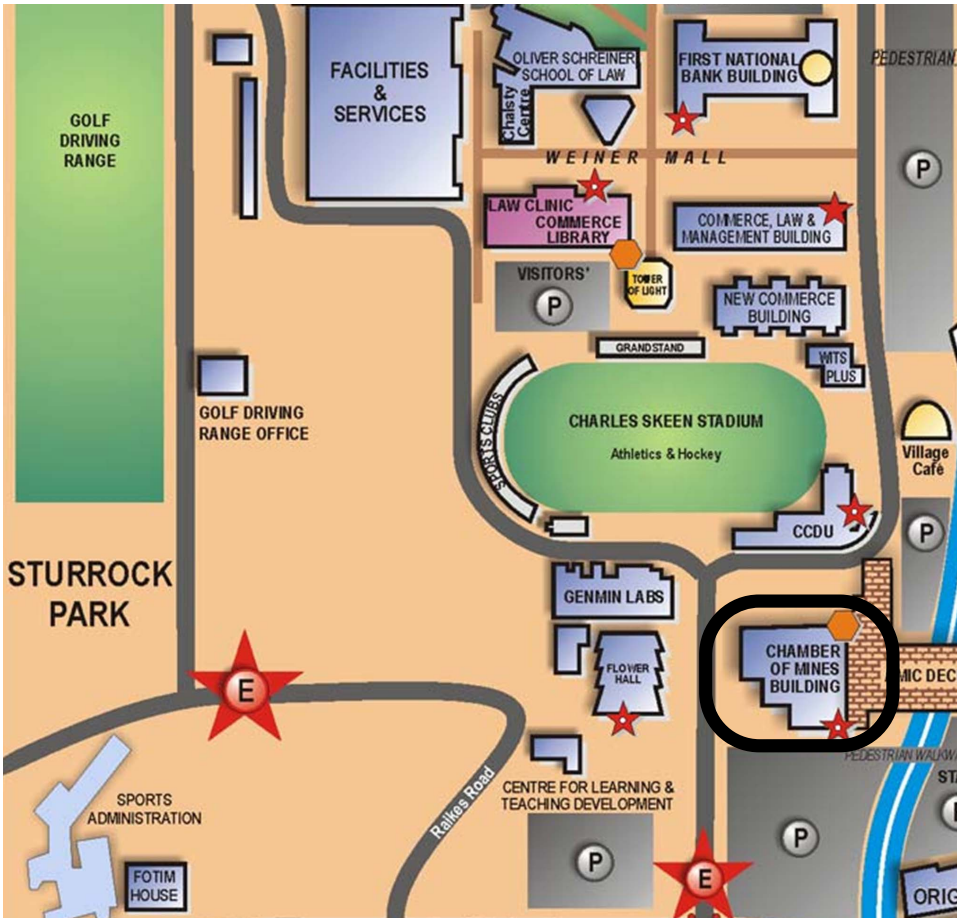


Figure D.25: Map of West Campus at the University showing the area the load profile was taken from, for the Chamber of Mines test.

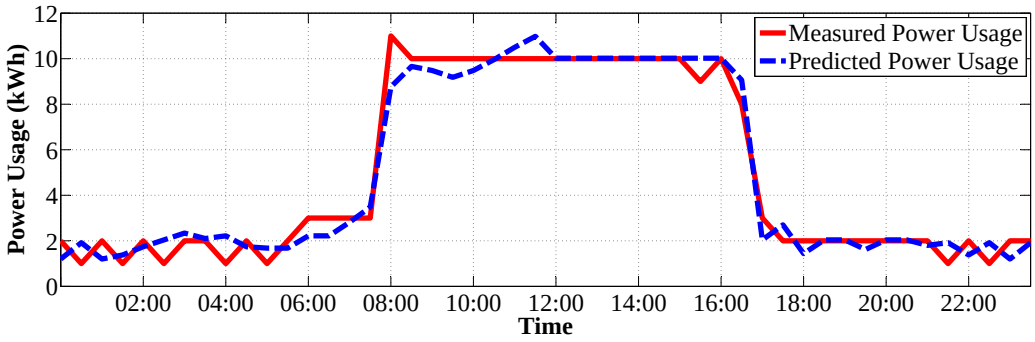


Figure D.26: Normalised load profile and the predicted power usage for a wing of a single floor of the Chamber of Mines building at the University for a general Monday.

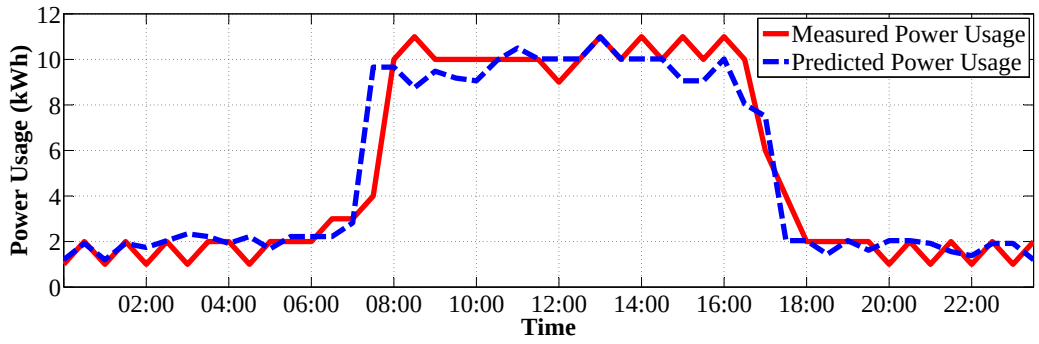


Figure D.27: Normalised load profile and the predicted power usage for a wing of a single floor of the Chamber of Mines building at the University for a general Tuesday.

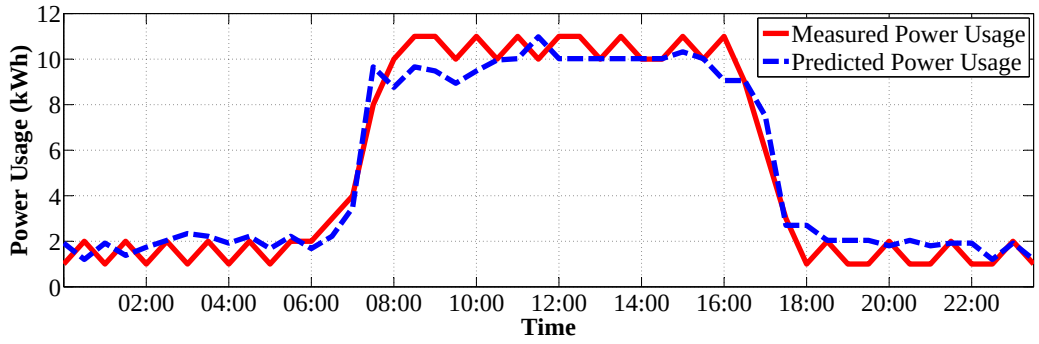


Figure D.28: Normalised load profile and the predicted power usage for a wing of a single floor of the Chamber of Mines building at the University for a general Wednesday.

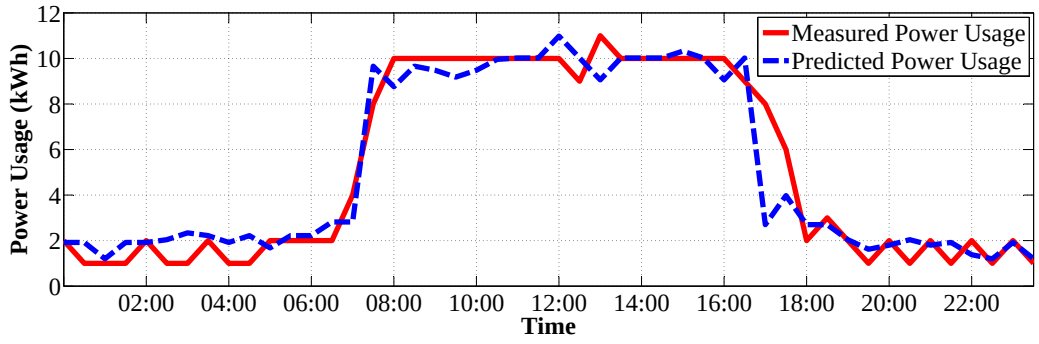


Figure D.29: Normalised load profile and the predicted power usage for a wing of a single floor of the Chamber of Mines building at the University for a general Thursday.

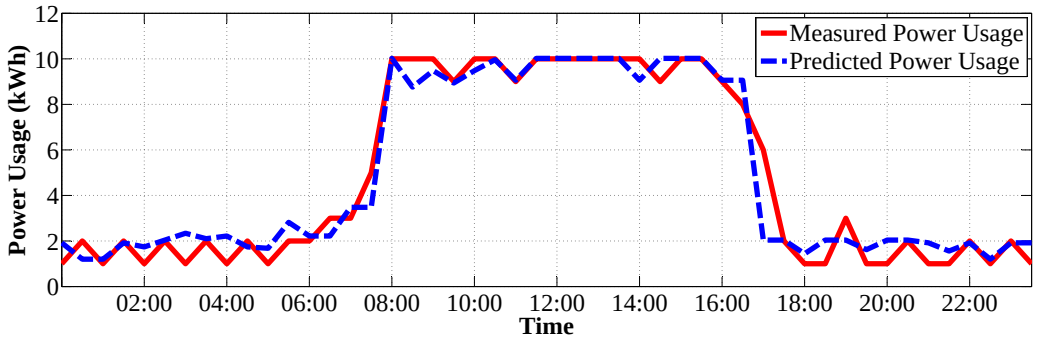


Figure D.30: Normalised load profile and the predicted power usage for a wing of a single floor of the Chamber of Mines building at the University for a general Friday.

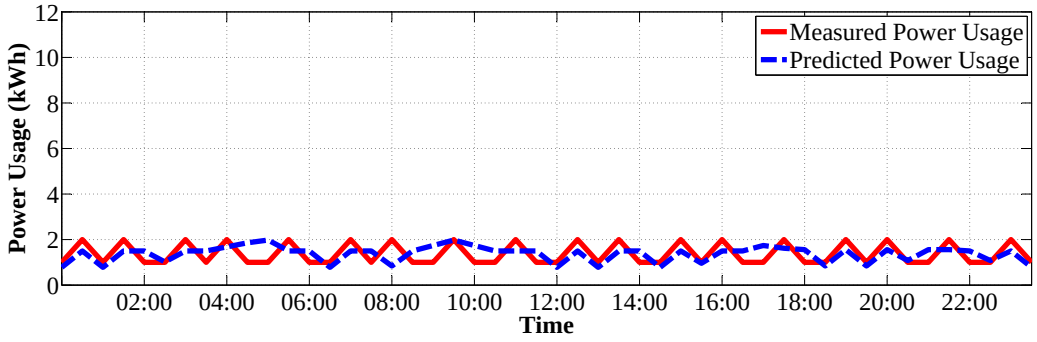


Figure D.31: Normalised load profile and the predicted power usage for a wing of a single floor of the Chamber of Mines building at the University for a general Saturday.

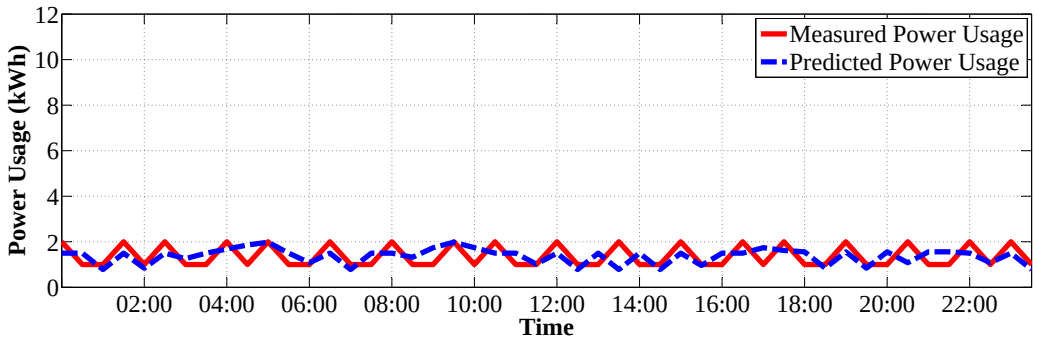


Figure D.32: Normalised load profile and the predicted power usage for a wing of a single floor of the Chamber of Mines building at the University for a general Sunday.

D.5 Case 5 - Single Plug Point with a Variable Load

Table D.1: Components connected for the single plug point test measurements.

Components
Base Load
Computer (for data recording)
Digital Power Meter
Variable Load
Coffee Machine

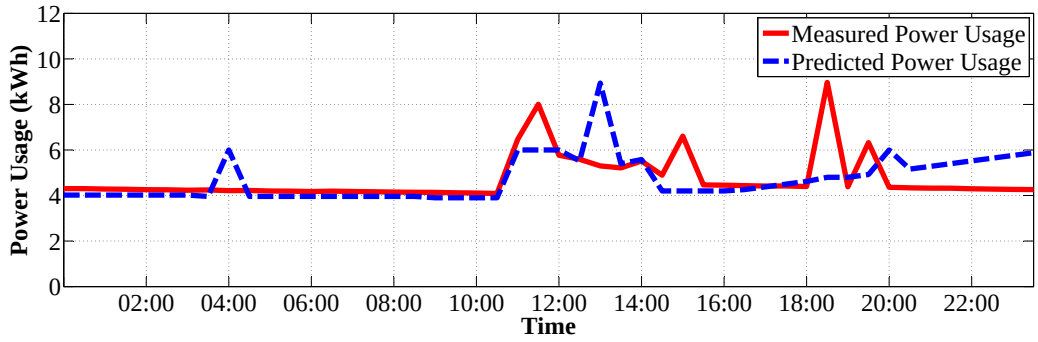


Figure D.33: Normalised load profile and the predicted power usage for a single plug point with a variable load for a general Monday.

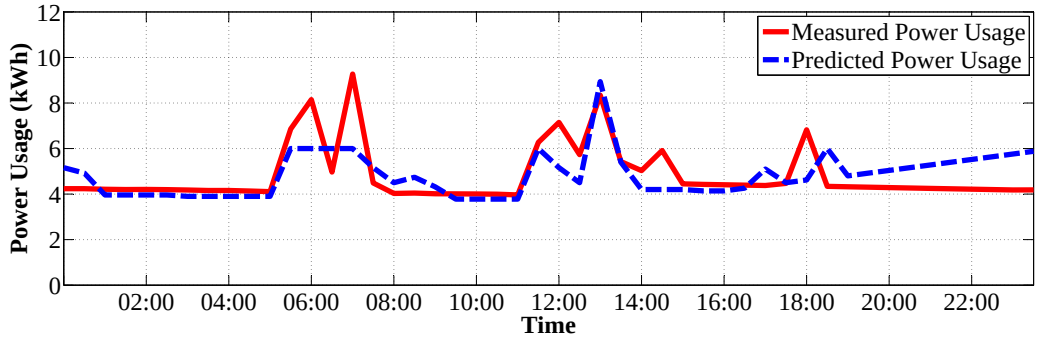


Figure D.34: Normalised load profile and the predicted power usage for a single plug point with a variable load for a general Tuesday.

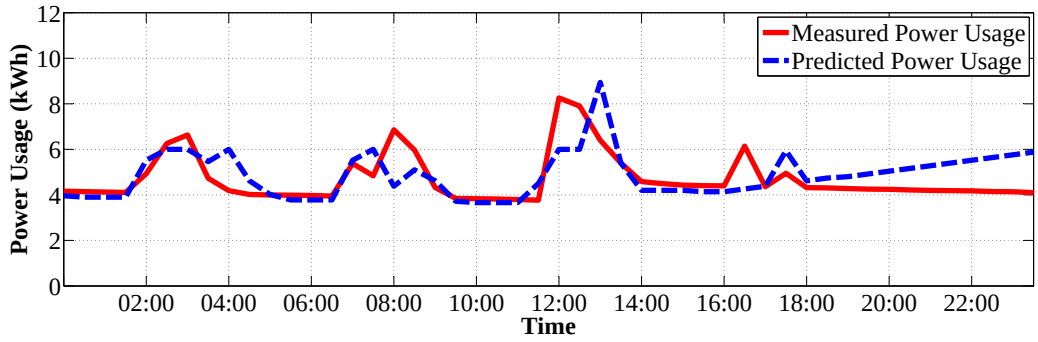


Figure D.35: Normalised load profile and the predicted power usage for a single plug point with a variable load for a general Wednesday.

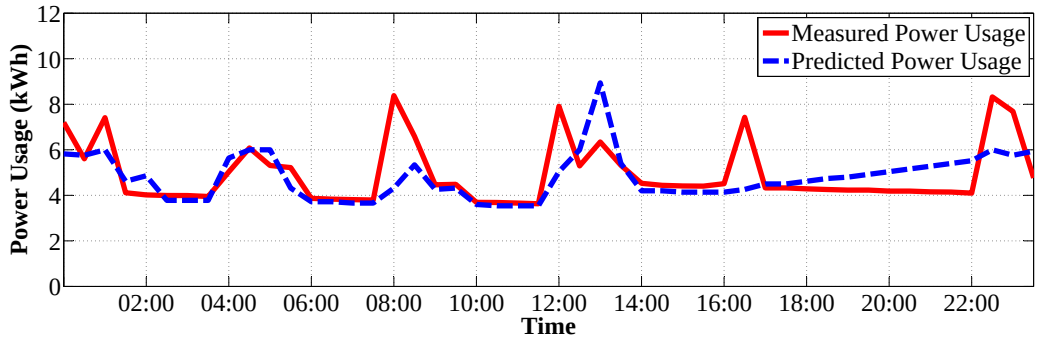


Figure D.36: Normalised load profile and the predicted power usage for a single plug point with a variable load for a general Thursday.

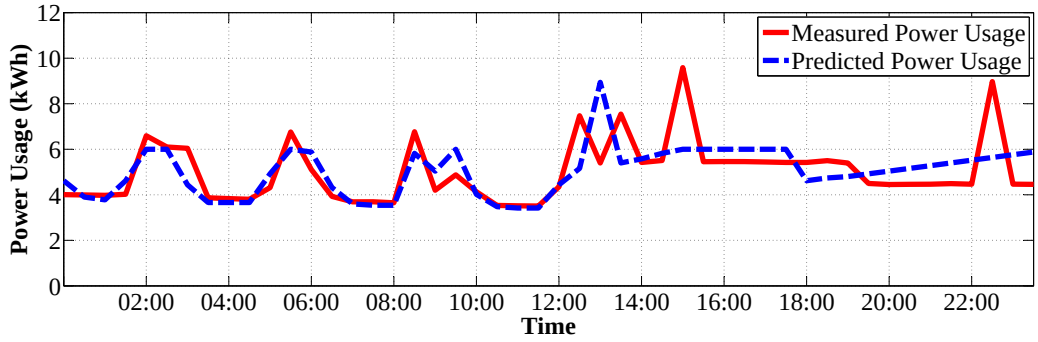


Figure D.37: Normalised load profile and the predicted power usage for a single plug point with a variable load for a general Friday.

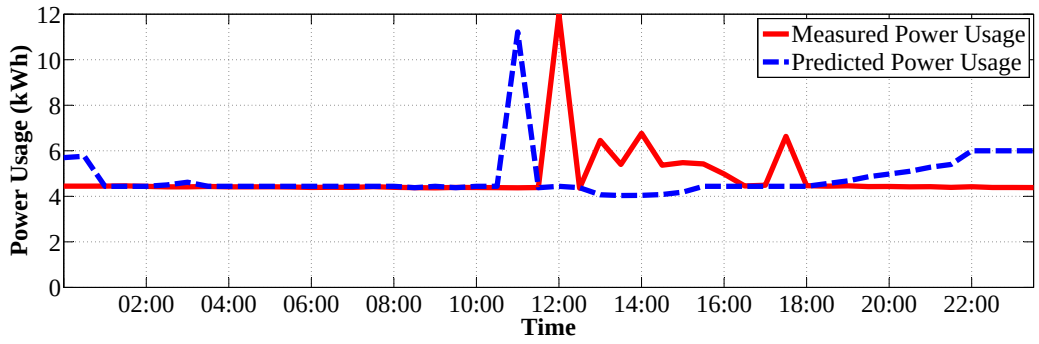


Figure D.38: Normalised load profile and the predicted power usage for a single plug point with a variable load for a general Saturday.

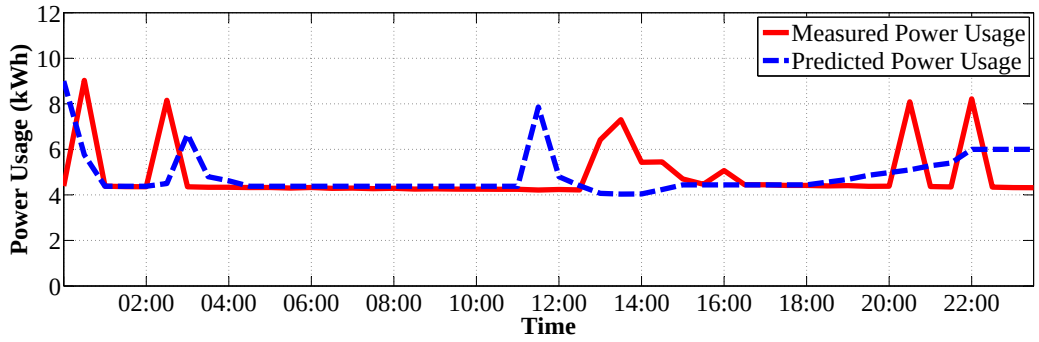


Figure D.39: Normalised load profile and the predicted power usage for a single plug point with a variable load for a general Sunday.

References

[1] S. E. Collier, "Ten steps to a smarter grid," in *Rural Electric Power Conference. IEEE*, pp. B2–B2–7, April 2009.

[2] X. Wei, Z. Yu-Hui, and Z. Jie-lin, "Energy-efficient distribution in smart grid," in *International Conference on Sustainable Power Generation and Supply 2009*, pp. 1–6, March 2009.

[3] A. Mohd, E. Ortjohann, A. Schmelter, N. Hamsic, and D. Morton, "Challenges in integrating distributed energy storage systems into future smart grid," in *IEEE International Symposium on Industrial Electronics*, pp. 1627–1632, July 2008.

[4] Christy van der Merwe (November 2007), "Blackouts to continue for up to 7 years - Eskom," *http://www.engineeringnews.co.za/article/blackouts-to-continue-for-up-to-7-years-eskom-2007-11-15*, Last accessed: 11 November 2011.

[5] James Macharia (April 2009), "South Africa still facing major energy crisis," *http://www.reuters.com/article/idUSTRE5315U320090402*, Last Accessed: 11 November 2011.

[6] R. E. Brown, "Impact of smart grid on distribution system design," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, IEEE*, pp. 1–4, July 2008.

[7] H. Farhangi, "The path of the smart grid," *Power and Energy Magazine, IEEE*, vol. 8, pp. 18–28, January-February 2010.

[8] J. Van Gorp, "Enterprising energy management," *IEEE Power and Energy Magazine*, vol. 2, pp. 59 – 63, January - February 2004.

[9] J. Park, Y. Park, and K. Lee, "Composite modeling for adaptive short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 6, pp. 450–457, May 1991.

[10] G. Gross and F. Galiana, "Short-term load forecasting," *Proceedings of the IEEE*, vol. 75, pp. 1558 – 1573, December 1987.

[11] D. Park, M. El-Sharkawi, I. Marks, R.J., L. Atlas, and M. Damborg, "Electric load forecasting using an artificial neural network," *IEEE Transactions on Power Systems*, vol. 6, pp. 442–449, May 1991.

[12] J. chang Lu, X. Zhang, and W. Sun, "A real-time adaptive forecasting algorithm for electric power load," in *Transmisson and Distribution Conference and Exhibition: Asia and Pacific, IEEE/PES*, pp. 1–5, 2005.

[13] NIST/SEMATECH (June 2010), "Box-Jenkins Models," *http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc445.htm*, Last accessed: 11 November 2011.

[14] T. Wonnacott and R. Wonnacott, *Regression: a second course in statistics*. New York: John Wiley & Sons, 1981.

[15] I. Moghram and S. Rahman, "Analysis and evaluation of five short-term load forecasting techniques," *IEEE Transactions on Power Systems*, vol. 4, pp. 1484 – 1491, November 1989.

[16] T. Haida and S. Muto, "Regression based peak load forecasting using a transformation technique," *IEEE Transactions on Power Systems*, vol. 9, pp. 1788 – 1794, November 1994.

[17] B. Kumar, J. Shanmugam, S. Janarthanan, and R. Santhiseela, "Development of expert system for the design of airborne equipment," in *The 23rd Digital Avionics Systems Conference*, vol. 2, pp. 6.C.4 – 61–10 Vol.2, October 2004.

[18] S. Rahman and R. Bhatnagar, "An expert system based algorithm for short term load forecasting," *IEEE Transactions on Power Systems*, vol. 3, pp. 392–399, May 1988.

[19] S. Klampfer, G. Globacnik, J. Mohorko, and Z. Cucej, "Expert systems design for automatic analysis of communication network simulation results," in *15th International Conference on Systems, Signals and Image Processing*, pp. 37 –40, June 2008.

[20] D. B. Fogel, *Evolutionary Computation: Toward a new philosophy of machine intelligence*. New York: IEEE Press, 2nd ed. ed., 2000.

[21] M. Bosque, *Understanding 99% of Artificial Neural Networks: Introduction & Tricks*. New York: Writers Club Press, 2002.

[22] R. S. Burns, *Advanced Control Engineering*. Oxford: Butterworth-Heinemann, 2001.

[23] T. Senjyu, H. Takara, K. Uezato, and T. Funabashi, "One-hour-ahead load forecasting using neural network," *IEEE Transactions on Power Systems*, vol. 17, pp. 113 – 118, February 2002.

[24] S. J. Kiartzis, A. G. Bakirtzis, and V. Petridis, "Short-term load forecasting using neural networks," *Electric Power Systems Research*, vol. 33, no. 1, pp. 1 – 6, 1995.

[25] C. hsien Kung, M. Devaney, C. min Huang, and C. ming Kung, "An adaptive power system load forecasting scheme using a genetic algorithm embedded neural network," in *Instrumentation and Measurement Technology Conference, 1998. IMTC/98. Conference Proceedings. IEEE*, vol. 1, pp. 308 – 311, May 1998.

[26] C. Lee, "Fuzzy logic in control systems: fuzzy logic controller. i & ii," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, pp. 404–435, March/April 1990.

[27] L.-X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*. New Jersey: PTR Prentice Hall, 1994.

[28] S. Sachdeva and C. Verma, "Load forecasting using fuzzy methods," in *Joint International Conference on Power System Technology and IEEE Power India Conference*, pp. 1–4, October 2008.

[29] R.-H. Liang and C.-C. Cheng, "Combined regression-fuzzy approach for short-term load forecasting," *IEE Proceedings - Generation, Transmission and Distribution*, vol. 147, pp. 261 – 266, July 2000.

[30] D. Srinivasan, C. Chang, and S. S. Tan, "One-day ahead electric load forecasting with hybrid fuzzy-neural networks," in *Biennial Conference of the North American Fuzzy Information Processing Society*, pp. 160–163, 19-22 1996.

[31] D. Srinivasan and M. Lee, "Survey of hybrid fuzzy neural approaches to electric load forecasting," in *IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century.*, vol. 5, pp. 4004 – 4008, October 1995.

[32] G.-C. Liao, "An evolutionary fuzzy neural network approach for short-term electric power load forecasting," in *Power Engineering Society General Meeting, 2006. IEEE*, 2006.

[33] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press, 1975.

[34] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Massachusetts: Addison-Wesley, 1989.

[35] C.-H. Chang and Y.-C. Wu, "The genetic algorithm based tuning method for symmetric membership functions of fuzzy logic control systems," in *International IEEE/IAS Conference on Industrial Automation and Control: Emerging Technologies*, pp. 421 –428, 1995.

[36] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," vol. 3, pp. 95–99, Kluwer Academic Publishers, October 1988.

[37] Y. H. Pengfei Guo, Xuezhi Wang, "The enhanced genetic algorithms for the optimization design," in *3rd International Conference on Biomedical Engineering and Informatics*, pp. 2990 – 2994, 2010.

[38] University of the Witwatersrand (2008), "Barnato Residence History," *http://student.wits.ac.za/Residences/Barnato/Barnato+History.htm*, Last Accessed: 24 October 2011.

[39] C. Asbury, "Weather load model for electric demand and energy forecasting," *IEEE Transactions on Power Apparatus and Systems*, vol. 94, pp. 1111–1116, July 1975.

[40] N. Amjady, "Short-term hourly load forecasting using time-series modeling with peak load estimation capability," *IEEE Transactions on Power Systems*, vol. 16, pp. 798 – 805, November 2001.

[41] K.-L. Ho, Y.-Y. Hsu, C.-F. Chen, T.-E. Lee, C.-C. Liang, T.-S. Lai, and K.-K. Chen, "Short term load forecasting of taiwan power system using a knowledge-based expert system," *IEEE Transactions on Power Systems*, vol. 5, pp. 1214–1221, November 1990.

[42] T. Senjyu, S. Higa, and K. Uezato, "Future load curve shaping based on similarity using fuzzy logic approach," *IEEE Proceedings - Generation, Transmission and Distribution*, vol. 145, pp. 375–380, July 1998.

[43] C. Negoită, *Fuzzy Systems*. Cybernetics ans systems series, Kent: Abacus Press, 1981.

[44] E. Scheffer, *Bridging the gap between Fuzzy Logic and Mathematical Modelling*. PhD thesis, Faculty of Engineering, University of the Witwatersrand, Johannesburg, 2001.

[45] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*. New Jersey: Wiley-Interscience, 2004.

[46] K. Man, K. Tang, S. Kwong, and W. Halang, *Genetic Algorithms for Control and Signal Processing*. Great Britain: Springer, 1997.

[47] University of the Witwatersrand (2011), "East Campus—Maps," *http://www.wits.ac.za/maps/606/east_campus.html*, Last accessed: 11 November 2011.

[48] University of the Witwatersrand (2011), "West Campus—Maps," *http://www.wits.ac.za/contactwits/maps/westcampus.htm/609/west_campus.html*, Last accessed: 11 November 2011.