Evolutionary Combinatorial Optimisation for Energy Storage Scheduling, and Web-based Power Systems Analysis using PHP

By

Simon Unekwu Agamah

Thesis submitted for the degree of

Doctor of Philosophy

School of Mathematics, Computer Science, and Engineering

City, University of London

September 2016

Table of Contents

List of Figures

List of Tables

List of Symbols

Evolutionary Combinatorial Optimisation Formulation

i,j	Index variables
t	Time step duration
t_i	Time interval i
D_i	Demand in interval i
D_p	Peak demand
F_1	Peak shaving objective
D_L	Peak-trough Demand margin
F_2	Load-levelling objective
b	Number of bins
E	Energy Storage System Capacity
P	Energy Storage System rated power
μ	Energy Storage System efficiency
C	Network firm capacity
K	Finite set of numbers to be searched for a target sum
k_i	Element i in set K
$s(k_i)$	Integer representing size of element k_i
m	Target sum
x_i	Coefficient for element size to show if element k_i is included in solution (1 or 0)
S	Schedule string in Genetic Algorithm
$Sia:b$	Schedule string i from point a to b
O	Offspring schedule string
ε	Tolerance for subset sum cardinality

Power Systems Studies theory

V	Voltage magnitude
δ	Voltage Angle
P	Real power
Q	Reactive power
I	Current
Y	Admittance Matrix
y_{ij}	Element ij of admittance Matrix
Z	Impedance Matrix
z_{ij}	Element ij of impedance Matrix
θ	Angle of polar form admittance in radians
ε	Tolerance for Newton-Raphson solution
S	Apparent power
X	Reactance

List of Abbreviations

ANM	Active Network Management
BESS	Battery Energy Storage System
BF	Best Fit
BP	Bin-Packing
BPA	Bin Packing Algorithm
CAES	Compressed Air Energy Storage
CO	Combinatorial Optimisation
COM	Component Object Model
DG	Distributed Generation
DLC	Direct Load Control
DNO	Distribution Network Operator
DP	Demand Profile
DR	Demand Response
DSR	Demand Side Response
ESS	Energy Storage System
EV	Electric Vehicle
FF	First Fit
GACO	Genetic Algorithm Combinatorial Optimisation
GCP	Grid Connection Point
HHVM	Hiphop Virtual Machine
HTML	Hypertext Mark-Up Language
HTTP	Hyper Text Transfer Protocol
HV	High Voltage
JSP	Java Server Pages
KCL	Kirchoff’s Currenbt Law
KVL	Kirchoffs Voltahe Law
LV	Low Voltage
MV	Medium Voltage

NF	Next Fit
NR	Newton Raphson
PHP	PHP: Hypertext Pre-Processor
PHS	Pumped Hydro Storage
PSA	Power Systems Analysis
PU	Per-Unit
SDK	Software Development Kit
SNG	Synthetic Natural Gas
SOC	State Of Charge
SONI	Systems Operator Northern Ireland
SPC	Set-Point Control
SS	Subset-Sum
SSP	Subset Sum Problem
STOR	Short Term Operating Reserve
TSO	Transmission System Operator
WBPSA	Web-Based Power Systems Analysis
WBS	Web Based Simulation
WF	Worst Fit
WWW	World-Wide Web
XML	Extensible Mark-Up Language

Acknowledgements

Several people have contributed to this thesis through their support, guidance, and unshaking belief in my ability to handle the stresses that come with studying and working simultaneously. I would like to express my gratitude to them.

My parents, Engr. Simon Onu Agamah and Mrs. Christiana Amichi Agamah, have fully sponsored my education up to this advanced level. Without their financial support the completion of this thesis would not be possible. I cannot thank you enough.

My supervisor, Dr. Lambros Ekonomou, has been a mentor and a guide throughout the time I've spent as a student in City University. His experience, motivation, and supervision made the burden of research work much lighter. It has been a pleasure to work with you.

My colleagues at PanicGuard have always been supportive of my efforts during my education and part-time work at the company. I sincerely appreciate your encouragement.

My utmost gratitude goes to my family – my parents, and my siblings Stephanie, Elva and Valentine for their love and moral support throughout the time I was working on my research and getting used to life in London. Most especially to Valentine who always lent a listening ear and encouraged me whenever I complained about the struggles of working and studying. You have been a brother indeed and I am truly grateful.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning. The author hereby grants powers of discretion to the City University London librarian to allow this thesis to be copied in whole or in part without further reference to the author. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

Abstract

Two research areas are covered in this thesis: the formulation of a novel evolutionary combinatorial optimisation algorithm for energy storage system (ESS) scheduling, and web-based power systems analysis (WBPSA) using PHP programming. An increase in electricity demand usually calls for reinforcement of the network equipment to handle the new load and network operators sometimes postpone or avoid this reinforcement by using ESS to store electrical energy when network usage is low and release it to be used in the grid during periods of high demand. The ESS operation must be scheduled to be effective and there are several scheduling methods that depend on energy generation data, flexible time-of-use tariffs or closed loop set-points. This thesis proposes a method that uses only historic or forecasted demand data which is also a requirement for other methods. The methodology formulates an electricity demand profile and ESS as a combination of the one-dimensional bin packing problem and the subset sum problem and solves them heuristically with specific modifications and transformations to obtain viable schedules. The schedules may then be optimised further using genetic algorithm optimisation. Comparative analyses with other algorithms and case studies using real-world data are used for verification. The algorithm is shown to be effective and has some advantages when compared to other existing algorithms; hence it can be used in scenarios where other methods are not applicable. On the second topic the thesis explores web-based power systems analysis platforms and shows that most use a web server primarily as an interface for exchanging requests and results between a front-end web browser and specialised back-end computation software written in a general programming language. A web server runs programs written in scripting languages such as PHP, which is the most popular web server programming language. Recent versions of web scripting languages have the computational capabilities required for power systems analysis and can handle the task of modelling networks and analysing them. This provides an opportunity for a slimmer 2-tier framework in which the web server also acts as the computation layer. The requirements for general power systems modelling are discussed and a methodology for realising web-based simulation using PHP programming is developed. Some of the modelling functions are handled natively in PHP and some require the use of extensions. The results show that using PHP for simulations can result in simpler access to power systems analysis functions in websites and web applications. The memory consumed by the PHP library developed is seen to be low and the computation time for reasonably large networks is in the millisecond range.

1 Introduction

1.1 Background

1.1.1 Background: Energy Storage System Scheduling for Peak shaving and load-levelling

Recent forecasts suggest that the electricity demand of regions like the UK are expected to double by 2050 due to the uptake of new electrical loads such as electric vehicles (EVs), heat pumps which convert energy from the ground or air to heat, home electronics such as mobile phones, robots and other forms of demand growth such as construction of multi-use buildings with high electricity demand [1] [2]. In other regions such as developing countries there is also demand growth as mobile phones and other electrical appliances become more ubiquitous in electricity supply networks which are not as resilient as the networks in developed countries. This growth in demand presents challenges to electricity network operators at all levels of the networks in these regions. Even without the growth in demand the network operators face challenges in maintaining the networks within financial and technical constraints [3].

Electricity networks consist of three operational levels: Generation, Transmission and Distribution. The main parameter used to distinguish these different levels is the voltage at which the electric power is transmitted [3]. Electrical energy is first generated by converting other forms of energy such as chemical, mechanical or heat energy to electricity by Generator operators at a medium voltage. Using transformers the voltage is increased or “stepped-up” to a high voltage (HV) to minimise energy losses in transmission lines and cables as the power is transferred to consumers and load centres. The Transmission System Operator (TSO) is responsible for maintaining the frequency of the electrical power within defined tolerance limits by balancing the demand from the load centres and the supply from the generators using a number of balancing services [4]. TSOs are also responsible for the electricity transmission equipment at the HV level. Electrical equipment at load centres and consumer premises usually do not operate at the HV level and therefore the electrical power is passed through a different set of “step-down” transformers before entering the medium voltage (MV) and low voltage (LV) networks of industrial, transport and residential end-users which is known as the distribution network. Distribution Network Operators (DNO) have an obligation to maintain the safety and reliability of LV and MV networks, connect new customers, and to extend and upgrade existing networks to meet changing needs [5] and are therefore responsible for keeping distribution lines and equipment operating within their maximum demand (or peak demand) limits which is referred to as their “firm capacity” to avoid power cuts and system blackouts.

To meet the demand growth generators may simply ramp up their output or increase their generation capacity, however when using conventional methods and fossil fuels this increase in generation is directly associated with an increase in CO_2 emissions. In the context of environmental concerns in more developed countries and regions like Europe with low carbon targets [1][6] the conventional approach will not be suitable.

The TSOs and DNOs also face challenges as a result of the continuous increase in demand exceeding installed firm capacity and network congestion as a result of customers demanding electricity concurrently during peak periods – the most relatable peak period being when several customers are getting ready for work and school in the mornings or returning from work in the evenings

The conventional approach to resolving these peak demand violations of network capacity is to reinforce the network by including more lines and transformers or replacing equipment to have higher tolerance ratings. In the worst cases as is prevalent in developing countries the problem is resolved by simply taking some customers off the network intermittently in an exercise known as “load shedding” or “rolling blackouts” [7][8]. The problem with conventional reinforcement is that it can be costly in terms of finances and logistics – the operators will need to buy new equipment, dig up new tunnels for cables, etc. and this upgrade may also lead to addition of underutilised capacity. In most cases the demand increases only in terms of the peak and not in terms of the total energy demand, which makes it difficult to justify network reinforcement in economic terms [9]. Load shedding is the worst option as it leads to widespread customer dissatisfaction and utility companies not fulfilling their obligations.

For these reasons cost-effective and efficient methods of resolving these issues must be explored to make efficient use of existing generation and network capacity. One of the solutions that is proposed by several reports and industry experts is the use of energy storage systems in the operation of the electricity networks as a tool for reshaping demand [1], [2], [9], [10]. Electrical energy storage systems (ESS) are devices or systems that convert electrical energy to a form that can be stored and then converted back to electrical energy when it is required [3].

ESS are currently in use for different applications as indicated in [11] including power quality improvement, electricity supply reserve, backup power for protection devices in electrical networks and also to support the increased penetration of renewable energy sources connected to the grid as distributed generation (DG). The uses of ESS that are related to the problem regarding peak demands exceeding network capacity are peak demand reduction (or shaving), load levelling and demand time shifting. These concepts are identifiable from their names; peak demand reduction means reducing the magnitude of peak demand, load levelling refers to flattening the demand profile to reduce the margins between peak and trough demands thereby making the demand more predictable and improving the efficiency of generators, and demand time shifting means moving the time a certain magnitude of demand occurs to a different time in a given period.

Energy Storage Systems may be used to shift loads to periods of lower demand and reduce the peak demand by storing energy when the demand on the network is low and releasing the stored energy for use when the demand is high. Customers may also use ESS to reduce their energy bills in markets with flexible tariffs and incentives such that electricity is less expensive during off-peak periods thereby providing benefits for the network operators and the customers.

One of the main challenges posed by the inclusion of ESS in an electrical system is the violation of constraints inherent to the system. Charging the ESS may cause overloading of equipment and lines, thereby violating thermal constraints and power limits; discharging them may cause over-voltage faults [12]. There are also limitations in storage capacity, and the strong link between times of energy storage and availability of energy at other times – also referred to as the *inter-temporal* nature of storage [12] which simply means energy cannot be used from ESS unless it was stored previously. Thus the full benefits that can be obtained ESS may not be realized because their contribution will always be curtailed at certain points.

As a result of these challenges ESS operations have to be scheduled as part of Active Network Management (ANM) schemes to maximise the benefits that are obtainable. ANM refers to devices, systems and practices that operate pre-emptively to maintain networks within accepted operating parameters [13]. ANM that includes devices with inter-temporal effects remains an open problem [12] and to include ESS in a network or electrical system effectively a plan must be developed in the form of an algorithm to select the optimal future charging and discharging periods based on historic information or a forecast.

ANM is also enabled with the rise of the “smart grid” which may be described as the existing grid with a communications layer to improve visibility of usage at all levels. Previously network operators could only collect information at their primary and secondary substations from where electricity is distributed to customers [14]. With the use of smart meters they can collect real time information with increased granularity and that can improve forecasts, planning and operations.

The focus this research in the area of electrical network management and consumer electricity utilisation is in the application of energy storage systems to achieve peak shaving and load levelling. To achieve these concepts and effectively operate the network within limits a scheduling methodology must be applied. A novel scheduling methodology is developed and presented in this thesis.

1.1.2 Background: Web-based Power Systems Analysis using PHP

Engineers must follow logical steps to study a physical system; the first step is to define the model, which requires hypotheses and simplifications; the model is then formalized into equations, and finally, the equations have to be solved either in a closed form or numerically [15]. In the same manner the first stage of building an electrical network for the generation, transmission and distribution of electricity is the planning and design stage. The operation of the electrical network must be established by representing the physical model of the network using conceptual and mathematical models. Evaluations and analyses must be made to ascertain the future system performance, safety, reliability and scalability [16].

Using these mathematical models calculations may be performed to understand the manner in which components of the network will interact with each other in transient and steady states and the condition of the network if a fault should occur. The process of analysing the operation of a physical model without assembling it is known as Simulation, which is defined as the representation of the behaviour or characteristics of one system through the use of another system, especially a computer program designed for the purpose [17].

These calculations and analyses may be done completely on paper using mathematical principles however as the system size increases it becomes more difficult to maintain the accuracy of solutions and they also take longer. Computers have been used for decades to carry out these calculations using simulation and modelling software [18] and they have several advantages over humans in terms of speed, accuracy and the complexity of problems they can be used to solve.

The simulation software deployment on computers has traditionally been in the form of computer programs which are run on software installed on personal computers or on a local network of interconnected computers. With the growth of information and communications technology and the internet which connects computers in different geographical locations, web-based simulation (WBS), analysis and remote control via a web browser interface such as Microsoft Internet Explorer™, Google Chrome™, Mozilla Firefox™ is increasingly becoming relevant [19][20]. There are several advantages to providing software over the internet including ubiquitous access to applications, ease of maintenance and new licensing models for providing software-as-a-service (SaaS). A scenario where WBS is useful is in collaboration among teams with members in different locations; models may be accessed over the internet and modified real-time.

The traditional approach of deploying power systems simulation software using desktop applications is because of the features of the programming languages used to develop them. The programming languages used for building simulation software are powerful general purpose languages such as C++ and Java and scientific programming languages such as Matlab [15]. These programming languages were not designed to be used for creating web pages and web sites and as a result cannot be used for web-based simulation without using an interface.

The programming language used for developing the front end of websites which is used by visitors of websites through web browsers is HyperText Markup Language (HTML). HTML is used for creating static pages which are documents that are sent to the users' computers from remote computers known as "web servers" when the web server internet address is accessed from the web browser [21]. HTML is suitable for information that doesn't change, however to create dynamic pages that are updated according to changing events on the web server other languages such as PHP: Hypertext Preprocesser (PHP) [22] and Active Server Pages .NET (ASP.NET) [23] are used. These are programming languages that are used to write modular applications known as "scripts" which are not standalone programs but must be interpreted by software residing on the web server. They are therefore known as web server scripting languages.

To perform simulation over the web the remote computers usually have the simulation "engine" which carries out the calculations and analysis written in the general purpose programming language such as Java or C++ and communication between the simulation engine and the web user is done via the web server running the web server scripting language. When a request is made by the user to access the remote model, the request is processed through web server scripts and received in the simulation engine, and the results are sent back through the web server scripts to the user. This creates a 3-tier architecture consisting of web browser – web server – simulation engine [24].

The 3-tier architecture is used for web-based power systems simulation and analysis (WBPSA) because most of the simulation software packages are written in general programming languages – as the IEEE open source task force on power systems analysis software [25] indicates, and also because earlier versions of web server scripting languages did not have the mathematical functions and features for performing the types of calculations required for WBPSA.

As web and internet usage for delivering software has grown recent versions of web server scripting languages have also evolved to have features for general purpose programming [26]. They also have more add-ons and packages created by third party developers that are being used to extend the capabilities of the programming languages beyond their core functionality. This presents an opportunity to develop more complex applications in web server scripting languages and possibly reduce the need for a 3-tier architecture, which requires more server resources and has several points of failure, to a 2-tier architecture consisting of web browser – web server.

PHP is the most used web server scripting language [27], [28] and more recent versions of the language have tools that may be used for WBPSA such that it serves as the simulation engine without communicating with software written in a general purpose programming language. There are several advantages in using this approach including a slimmer architecture, fewer points of failure and ease of deployment in existing websites.

PHP has never previously been used for developing power systems simulation and analysis software and there are several challenges regarding the conversion of mathematical models to software models that can run accurately and efficiently in PHP. There are also challenges in identifying and providing functionality which is not available in the PHP core language but are requirements for power systems simulation and analysis. The structure of the PHP library for WBPSA, the programming methodology and rules guiding the interactions of internal components representing principles for power systems studies must be defined. This thesis presents a novel methodology developing a software library for web-based power systems analysis using PHP.

1.2 Thesis Objectives

Several methodologies and algorithms have been formulated and some are already in use for scheduling ESS in Active Network Management schemes at the distribution level or at the consumer level. The majority of these algorithms depend on generation prices, end-user time-of-use tariffs and energy prices. Such approaches, however, have failed to address the limitation in the data available to the distribution network operator (DNO) or consumer, and the amount of influence they have over these parameters. Furthermore, some of these approaches can also be computationally complex.

The data the DNO and consumer usually have full control over is limited to the demand profile information (forecast or historic) and the ESS parameters. However, the determination of ESS schedules with only these data is technically challenging because of the nonlinear nature of demand and constraints on ESS. As a result the options available for a versatile scheduling algorithm which is applicable to any network or profile are closed loop set-point control (SPC) which switches on storage at given demand set-points similar to a thermostat, and heuristic methods. SPC is usually based on real-time control and therefore does not take into account future events which may lead to suboptimal solutions globally; it is also vulnerable to short-term spikes in demand.

This indicates a need to provide simple but efficient heuristic options for DNOs and consumers to schedule storage effectively using the data available to them. Therefore, the objective of this thesis on the topic of ESS for peak shaving and load-levelling is as follows:

- **i.** Formulate a methodology or algorithm to generate optimal schedules for dispatching ESS at distribution network or consumer level for peak shaving and load-levelling using only the demand profile and ESS parameters.
- **ii.** Verify the algorithm by comparative analysis with other algorithms and in case studies with a focus on robustness, versatility, and adaptability to any network with these parameters provided.

On the subject of web-based Power Systems Analysis using PHP the key challenge is in identifying the concepts required for defining and modelling Electrical Systems and the availability of functions for these concepts in PHP. Because most studies in the field of WBPSA have only focused on 3-tier architectures there is a paucity in analytical tools and application libraries that can be deployed easily on a web server for PSA, and the use of PHP for this purpose has not been investigated.

Therefore on the research topic of Web based power systems analysis the objectives of this thesis are as follows:

- **i.** Identify the fundamental concepts required for general power systems analysis in any programming language and the availability of core functions or third-party libraries for modelling these concepts in PHP
- **ii.** Develop an application library for realizing general power systems analysis in a web-based 2-tier framework using PHP and benchmark its performance and accuracy using standard networks as test cases

1.3 Outline of the Thesis

In arranging the content it was of utmost importance to the author that the progression of ideas in both areas is recognisable and continuous to the reader. Therefore the overall structure of the thesis takes the form of eight chapters; the first chapter is this introduction which covers the background, objectives, outlines the structure, and the contribution of the work. The remaining sections are structured as follows:

The second and third chapters are on the literature and methodologies of ESS scheduling in previous works and the proposed algorithm;

- Chapter 2 presents a review of relevant literature and existing methodologies in ESS scheduling highlighting their strengths, vulnerabilities and gaps in research. The reviews will show the parameters required for optimisation in each method and provide a synopsis of their approach, assumptions and results.

- In Chapter 3 the problem of ESS scheduling is defined clearly by way of equations and illustrations as a combination of the bin-packing and subset sum problem for finding the best placement for energy storage in a demand profile. The constraints are shown and a proposed novel methodology for scheduling is formulated using heuristics such that several viable schedules are produced and then further optimised using Genetic Algorithm optimisation.

The fourth and fifth chapters are on the topic of Web-based Power Systems analysis;

- Chapter 4 reviews previous research literature in WBPSA, and software currently being used to deliver PSA on the web with emphasis on the programming languages used and their software architecture. It also covers the fundamental concepts and requirements for general power systems analysis focusing on power flow and short circuit studies.

- Chapter 5 covers the development of the PHP library for power systems analysis including the description of the functions for creating a model of a network and its elements as PHP objects, and the functions for executing load flow and short circuit power systems studies.

The sixth and seventh chapters are for tests and results of the formulated methodologies in the preceding chapters, and the final chapter presents the conclusions drawn from each topic.

- Chapter 6 presents the comparative analysis and case studies used to test and verify the combinatorial optimisation method for ESS scheduling. Four cases will be examined, in each case the base case of the demand profile is compared to the results after the algorithm has been applied. The research data in this thesis is drawn from three main sources: previous publications, project consultation documents for installation of Energy storage systems in an existing network, and a demand profile generator based on a model of historic energy consumption in a distribution network. The results are discussed at the end of each study.

- Chapter 7 presents tests carried out to verify the accuracy of the PHP power systems library and measure its performance. A number of standard test cases are used for the studies to confirm the accuracy of the results of computations. The computation time and memory required by the library for each of the major operations in the power systems studies are also measured. The impact of the operations on the computing resources is discussed.

- Chapter 8 presents the conclusions and further research recommendations starting with the ESS scheduling algorithm. The potential impact on the smart grid and the author's view on the future of storage are discussed. The conclusions from the development of the PHP power systems analysis library are also presented, the likely users of the library and the possibilities for further development in the area of web-based power systems analysis.

1.4 Contribution

The ESS scheduling algorithm for peak shaving and load-levelling will contribute to existing knowledge and practices in the following ways:

i. This study provides new insights in power systems algorithm design by extending the use of the bin-packing algorithm and subset sum algorithm to power systems studies. The bin-packing algorithm is used in other industries, for example in logistics for loading shipping containers, and in computing for optimising data storage on hard drives. This is the first time an electricity demand profile is being viewed as a set of distinct containers and energy storage as a sum of smaller items to be packed or removed from those containers. Any new development in these adapted algorithms will also imply an improvement in the performance of the ESS scheduling algorithm.

ii. Using this method Distribution Network Operators will have full control of the set of parameters required to integrate ESS in a regulated or deregulated market scenarios without compromising on effectiveness. This was not the case previously as more parameters and assumptions are required for other methods. The DNO may now apply this algorithm to generate schedules for Energy storage systems integration to keep networks operating within limits and maximise the utilisation of existing equipment to postpone costly network reinforcement.

iii. At the consumer level, the algorithm may be applied to flatten demand, reduce consumption during peak tariff periods and make savings in energy cost. By using this method the demand profile at the consumer level will also have less variation and cannot lead to new peak demands for the DNO to supply as a result of the time-of-use tariffs.

The PHP power systems library makes important contributions to power systems simulation in the following ways:

i. The library introduces slimmer architecture that requires fewer computational resources and has fewer points of failure. 3-tier architectures require computing environments to be configured for the simulation engine and for the web server. This library uses only the web server environment.

ii. The library will allow websites that have been built using PHP to embed power systems models for various purposes with significantly less difficulty. For example, a network model operating an experimental algorithm may be published directly to a PHP website and actively operated by peer reviewers to view how conditions of the network are affected by changing parameters. This will represent an improvement in results presentation as authors will no longer be confined to presenting results as static tables and charts and tables. To achieve this in the present form of WBPSA will require licensing fees for the commercial packages, significant programming skills and more internet computing resources for the simulation server.

1.5 List of Author's Publications

The following publications have produced based on the work presented in this Thesis.

Journal Publications

1. Simon U. Agamah, Lambros Ekonomou, A Heuristic Combinatorial Optimization Algorithm for Load-levelling and Peak Demand Reduction using Energy Storage Systems, International Journal of Energy and Environmental Engineering, IJEE-D-16-00331, 2016, Under Review.

2. Simon U. Agamah, Lambros Ekonomou, Energy Storage System Scheduling for Peak Demand Reduction using Evolutionary Combinatorial Optimisation, Energy Conversion and Management, ECM-D-16-04307, 2016, Under Review.
3. Agamah S., Ekonomou L., Peak demand shaving and load-levelling using a combination of bin packing and subset sum algorithms for electrical energy storage system scheduling, IET Science, Measurement and Technology, DOI 10.1049/iet-smt.2015.0218, 2016.

Conference Publications

1. Agamah S., Ekonomou L., A PHP application library for web-based power systems analysis, 9th IEEE European Modelling Symposium on Mathematical Modelling and Computer Simulation, Madrid, Spain, pp. 353-358, 2015.

Book Chapters

1. Agamah S., Ekonomou L., A methodology for web-based power systems simulation and analysis using PHP programming, Electricity Distribution, Energy Systems Series, Springer-Verlag Berlin Heidelberg, pp. 1-25, 2016, DOI 10.1007/978-3-662-49434-9_1.

2 Electrical Energy Storage Systems Management and Scheduling: Related work and Literature

2.1 Introduction

This section provides a background on Electrical Energy Storage Systems which are referred to as Energy Storage Systems (ESS) in the context of this thesis. The review will provide information on the types of Energy storage systems, roles and functions of energy storage systems and the current state of development. The current methods used to schedule the charging and discharging of ESS will be reviewed and their strengths and weaknesses will be assessed.

2.2 Electrical Energy Storage Systems Review – types, functions and roles

Electrical Energy storage systems (ESS) are devices or systems that convert electrical energy to a form that can be stored and then converted back to electrical energy when it is required [3]. Energy storage has been essential for providing reliable energy supplies for centuries, from stockpiles of coal and uranium to reservoirs of gas and water. Advances in material science allow for storing higher energy densities in forms that are rechargeable i.e. in a manner that the energy may be used and replenished up to a certain amount. Energy can be stored in several forms: thermal, chemical, gravitational mechanical, and in electromagnetic fields before being converted to its end use as electricity or heat [6].

Electrical Energy storage systems may be classified according to various parameters such as their technology, energy form, energy capacity, power density, duration of use, frequency of use and functions. Figure 2.1 shows a classification of electrical energy storage systems according to energy form as mechanical, electrochemical, thermal, chemical and electrical [11]. This shows the different ways in which electrical energy may be stored for use at a different time or location, and the various devices used in storing energy.

ESS capacity is defined in terms of the maximum amount of power that can be drawn from them or used to charge them, known as the rated power. The energy capacity is defined in terms of the amount of current that can be drawn from them over the period of an hour in Ampere-Hour unit (Ah) or the amount of power they can supply over the period of an hour in Kilowatt-Hour unit (kWh).

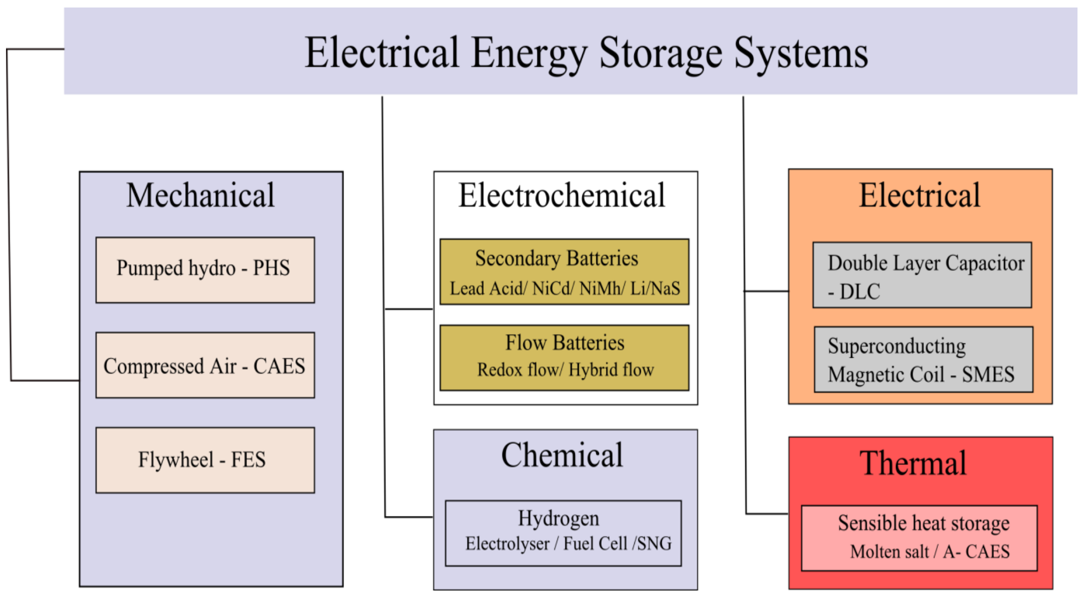


Figure 2.1. Electrical Energy storage systems classification by energy type

2.2.1 Roles of Energy Storage Systems

The roles played by ESS are largely dependent on a number of factors including the duration of use, frequency of use and the amount of energy they can store. The roles that are played by ESS may vary from the viewpoint of electricity network operators, consumers and electricity generators. Figure 2.2 shows how the frequency and duration of use impacts the different roles.

Electricity Network Operator roles for energy storage are as follows [11]:

1. **Time Shifting**

When electricity is supplied to consumers there is usually a gap between the amount of energy used during the periods of highest demand known as the peak periods, for example when people get ready for work and use their boilers around the same time, and the periods of lower demand known as off-peak periods, for example at night when users demand is low as most consumers are asleep. The peak usually increases annually and as a result the installed capacity of the distribution lines and other used to supply electricity to consumers must be reinforced. ESS allows electricity distribution network operators to store energy during off-peak periods and then use the energy during the peak periods to bridge the demand gap. This shift in the time of use of energy that is stored is known as time shifting. Efficient time shifting results in peak demand reduction and load-levelling, which is a “flattening” of demand across time periods.

2. **Power Quality**

Electricity utilities including transmission and distribution network operators have a mandate to maintain supply power voltage and frequency within a set tolerance. They maintain power frequency within the tolerance by adjusting the output of generators or changing the settings of equipment such as transformers, which control the ratio of voltages on the side of the supplier and the consumer. ESS may be used to provide these functions by either charging or discharging to increase the demand at the consumer side or reduce it from the viewpoint of the supplier.

3. **Network Congestion reduction to defer reinforcement**

Congestion on an electricity network occurs when many consumers demand electric power at the same time through the same equipment. If the electricity demand instantaneously exceeds a certain amount failure may occur. ESS may be used to postpone the reinforcement of network equipment to prevent failure. Utility companies may install ESS at appropriate substations and along with time shifting strategies postpone reinforcement of the network.

4. **Isolated grids**

Some locations do not have direct access to supply lines from the wider electricity grid and must therefore operate as islanded networks. ESS may be used to increase the penetration of renewable energy in such networks, for example in [29] ESS is used to increase the use of wind energy in the Shetlands, UK. ESS are also used to increase demand to match the supply of diesel generators on such islanded grids to provide stable electricity supply.

5. **Emergency power supply for protection and control equipment**

The set of equipment that provides protection for other equipment in electricity networks need a reliable power source. Batteries are used as an emergency power supply for the equipment in case of a power outage.

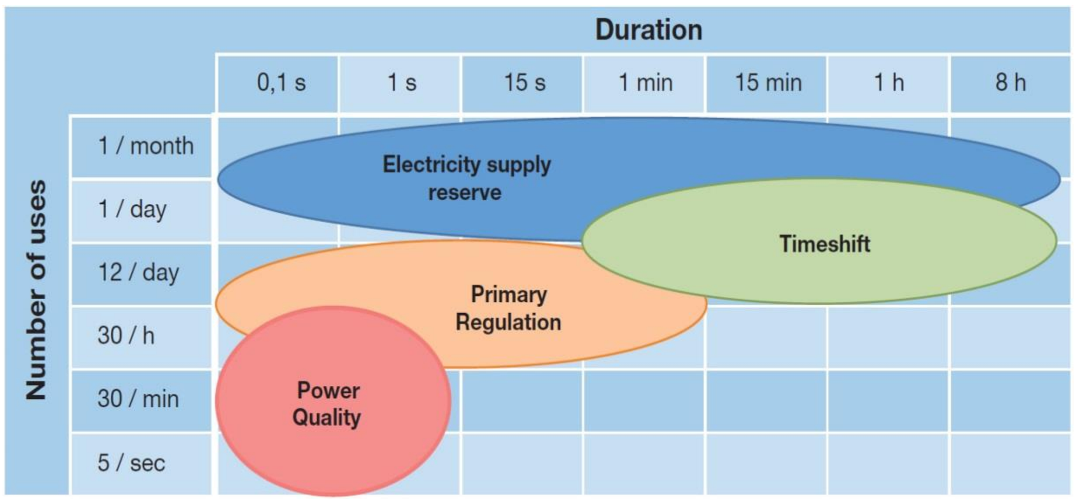


Figure 2.2. Roles of energy storage systems according to frequency and duration of use

The ESS used by utilities are usually large scale and may take up a lot of space. In some cases such as Pumped Hydro Storage (PHS) and Compressed Air Energy storage (CAES) the geographical features of the installation location are used as an advantage. PHS may take advantage of a mountainous terrain to pump water up to tanks on mountains during off-peak periods and run it down through turbines along the slope of mountainside to generate electricity during peak periods. Compressed air may be stored in caverns and used to run gas turbines for electricity generation. In the case of large scale battery energy storage they are usually installed in interconnected arrays with each unit being about the size of a shipping container. Figure 2.3 shows a section of a 10MW AES Advancion energy storage array in Ireland consisting of 53,000 batteries arranged in 136 nodes. The Kilroot array balances supply and demand and support the All Island transmission grid via System Operator Northern Ireland (SONI). The Advancion array also enhances power supply, enables more efficient dispatch of existing generation assets, and increase the ability to integrate renewable power sources [30].

Figure 2.3. Grid-scale battery storage array in Kilroot, Ireland

ESS are also used by consumers for different functions including time shifting, energy independence and backup power supply. Recently there has been an increase in the penetration of ESS in consumers' homes and business premises as a result of incentives such as the Feed-in-Tariff [31] in the UK where small-scale renewable generation is subsidised, and as a result encouraged the uptake of solar and wind generation in homes. The excess energy generated by these renewable sources is either exported or stored. Manufacturers and suppliers have also moved to provide small-scale energy storage to consumers, devices such as the Moixa Maslow™ [32] shown in Figure 2.4 are about the size of a boiler unit and can provide up to 10kWh of storage connected behind the electricity meter and usually also to a solar energy installation. Cost saving is also achieved by consumers taking advantage of flexible time-of-use tariffs where energy prices are lower during off-peak periods

The roles of energy storage systems for consumers are as follows [11]:

1. **Time shifting and Cost-savings**

Consumers may use energy storage to take advantage of time-of-use tariffs that price electricity differently for peak and off-peak periods. Consumers may store energy when it is less expensive during the off-peak periods and use it during the peak periods when it is more expensive from the supplier.

2. **Emergency power supply**

ESS may be used to provide backup power supply for equipment such as fire sprinklers and emergency lighting in the event of power failure. ESS are also useful in medical and manufacturing industries for processes that require constant power supply. In regions with weak power network infrastructure ESS may be used to provide electricity to appliances during short-term outages.

3. **Mobility and mobile appliances**

ESS are used in electric vehicles (EV) as sole power sources or in hybrid configurations alongside internal combustion engines. EV batteries are also expected to be used to supply power to households or back to the electricity grid in concepts known as vehicle-to-home (V2H) and vehicle-to-grid (V2G). The most common and widespread application of ESS is in mobile devices such as mobile phones, laptop computers and tablet computers. ESS provide power to these devices and enable consumers to use them for long periods without a wired connection to a power supply point.

Figure 2.4. A Maslow Smart Energy Storage household unit

2.2.2 Classification of Energy storage system technologies by functions

Based on the roles of energy storage described previously the main functions of ESS are:

i. Provision of uninterruptable power supply (UPS)

ii. Power quality management

iii. Transmission and distribution grid support

iv. Load shifting

v. Bulk power management

The existing technologies for ESS are able to provide these functions differently depending on their rated power, energy content and the nominal discharge time. Figure 2.5 shows the different existing technologies and the functions they are usually used for according to these factors. Most ESS are modular with the exception of Compressed Air Energy Storage and Pumped Hydro systems which are dependent on geographical factors, therefore a higher capacity may be realized by combining several units. The main restrictions on capacity for modular units will be economic i.e. cost per kW or cost per kWh [11].

Using discharge time to categorize the technologies, they may be classified into short discharge time, medium discharge time and long discharge time.

1. **Short discharge time – seconds to minutes, energy-to-power ratio less than 1**

These include Double-layer capacitors (DLC), superconducting magnetic energy storage (SMES), and flywheel energy storage (FES). These have a capacity of less than 1kWh for a system with a power of 1kW.

2. **Medium discharge time – minutes to hours , energy-to-power ratio of between 1 and 10**

These include FES, electrochemical ESS i.e. batteries including Lead-acid (LA), Lithium Ion (Li-ion) and Sodium Sulphur (NaS) batteries. Batteries have advantages in the kW – MW and kWh – MWh range compared to other technologies including high energy density, fast charging behaviour and long life. Their typical discharge times are up to several hours. A system of rated power 1kW may have between 1kWh to 10kWh capacity.

3. **Long discharge time – days to months, energy-to-power ratio of greater than 10**

These include hydrogen (H_2) and synthetic natural gas (SNG) systems.

It can be concluded that no single universal storage technology is superior to all other storage systems, and their suitability depends on the purpose. The focus of this thesis is in Time shifting applications at consumer level or distribution grid level, particularly in peak demand reduction and in load-levelling. Therefore the technologies that will be suitable will be in the medium discharge time category with an energy-to-power ratio of between 1 and 10. The current technologies that exist and are best suited for this purpose from Figure 2.5 are battery energy storage systems and high energy super capacitors.

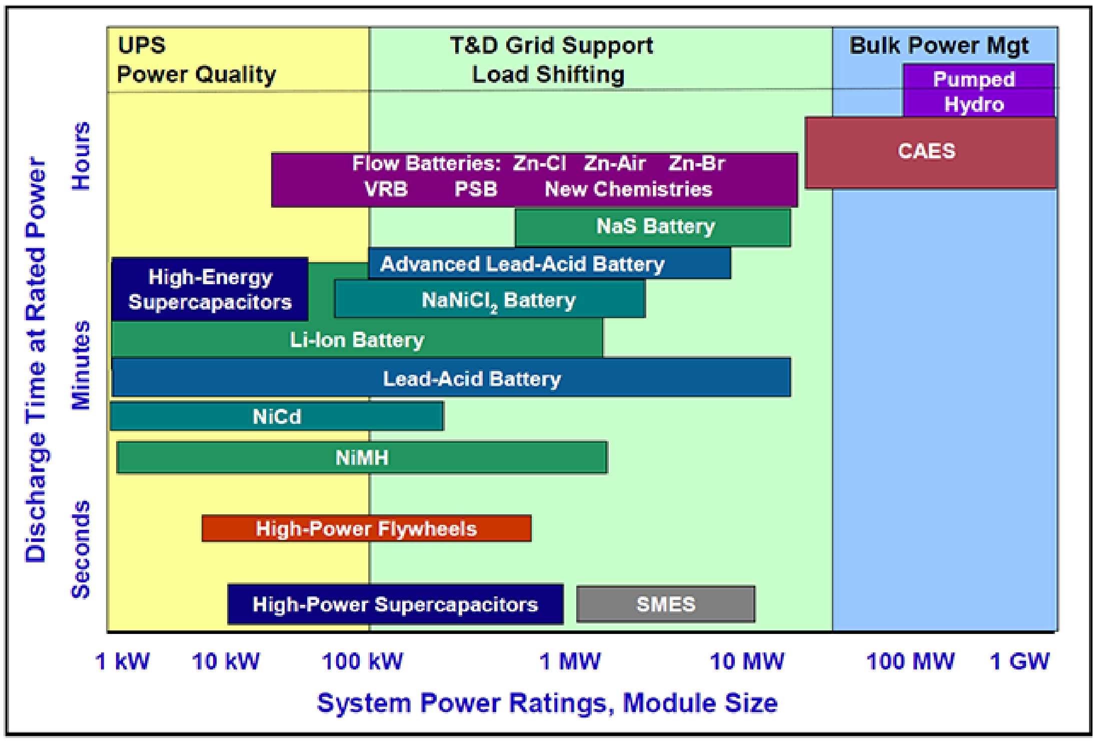


Figure 2.5. Categories of Energy Storage System technologies according to discharge time and system ratings

2.3 Peak shaving and Load-levelling functions of ESS

As stated in the roles of ESS, time shifting is used by network operators and consumers to manage network operation and to reduce the cost of energy bills using energy storage. Among the time-shifting operations to resolving problems surrounding network congestion and firm capacity limits are Peak shaving and load levelling. The benefits of peak shaving and load levelling include improved system reliability – as the system will always operate within limits, and also deferment of reinforcement when demand increases.

These related concepts are illustrated in Figure 2.6 – Figure 2.9 where the abscissa shows the time periods and the ordinate shows the magnitude of electric power demand measured at a load point. The effects of peak shaving and load levelling are shown as either reducing the demand from the original peak demand point (peak shaving) or increasing the lowest demands so they are closer to the peak demand and the margin is reduced (load-levelling) or performing both actions simultaneously (peak shaving and load-levelling).

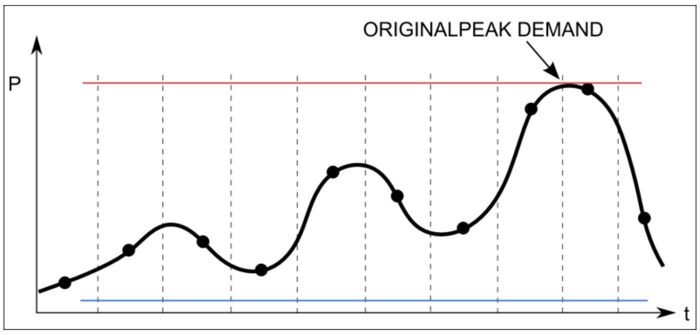


Figure 2.6. Original Demand Profile with no peak shaving and load levelling has wide margin between peak and trough demands

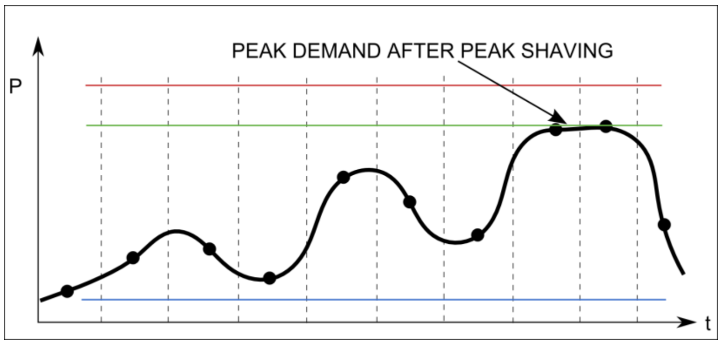


Figure 2.7. Demand Profile after Peak shaving without load-levelling showing lower peak demand

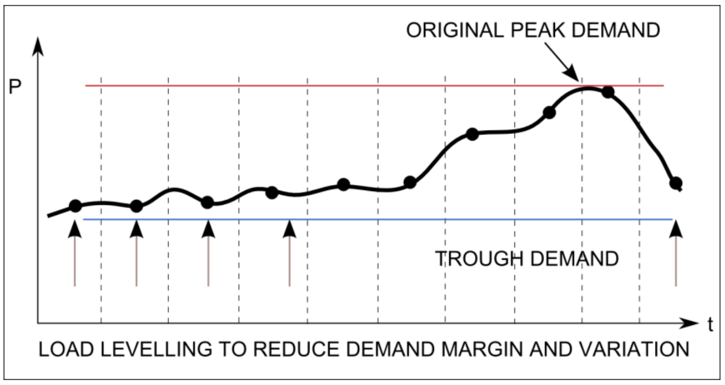


Figure 2.8. Demand profile after load levelling without peak shaving showing smaller peak-trough demand margin and less variation in demand

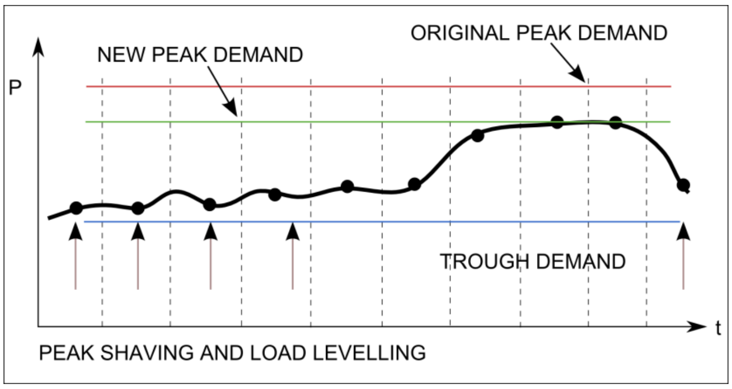


Figure 2.9. Demand profile after load levelling with peak shaving showing smaller peak-trough demand margin and less variation in demand

Peak shaving involves reducing the peak demand of a sub-network or a load point such as a consumer residence as observed from a point upstream on the network[33]. In the simplest form peak demand may be reduced by load-shedding. Load-shedding is an operation where network operators take some customers off the network during periods of peak demand leading to blackouts for those customers. This is the case in areas with weaker network infrastructure such as parts of sub-Saharan Africa [8].

In more developed areas with high priority on security of supply other arrangements including Short Term Operating Reserve (STOR) – generator companies are paid to come online to support the network for a short time or larger consumers are paid to take off some demand momentarily; Demand Response schemes (DR) – devices and consumers responding to higher peak prices; and Direct Load control schemes (DLC) – network operators turning off some deferrable loads during peak periods. Energy Storage Systems may be used to shift deferrable loads to periods of lower demand and reduce the peak demand.

Load-levelling is the reduction of variation in demand over a given time period by rescheduling loads to periods of lower demand or producing energy during those periods for storage and use during peak periods [33]. Load-levelling makes the demand profile more predictable and can also have the effect of peak demand reduction. A flatter demand profile with peak reduced increases the available capacity in the network. It also allows for efficient operation of generators.

The focus this research in the area of network congestion management and consumer electricity utilisation is in the application of Energy Storage systems to achieve peak shaving and load levelling.

2.4 Active Network Management and the need for Energy Storage System scheduling

One of the main challenges posed by the inclusion of ESS in an electrical system is the violation of constraints inherent to the system because they may act as loads or generators when charging or discharging. Charging the ESS may violate thermal constraints and power limits of the electrical network, discharging them may cause over-voltage faults [12]. There are also limitations in storage capacity, and the strong link between times of energy storage and availability of energy at other times – also referred to as the *inter-temporal* nature of storage [12], which simply means one cannot use energy that has not previously been stored. Thus the full benefits that can be obtained ESS may not be realized because their contribution will always be curtailed at certain points.

As a result of these challenges ESS is usually scheduled as part of Active Network Management (ANM) schemes to maximise the benefits that are obtainable. ANM refers to devices, systems and practices that operate pre-emptively to maintain networks within accepted operating parameters [13]. The first generation of ANM schemes implemented a monitor and control philosophy in which they used the measurement of network parameters as control signals for keeping parameters within limits. These operated in the order of several seconds to several tens of seconds. The second generation of ANM being developed go beyond simple monitor and control methods and incorporate advanced forecasting and scheduling [34].

Hence to include ESS in a network or electrical system effectively a schedule for operating it must be generated by a scheduling algorithm to select the optimal future charging and discharging periods based on historic information or a forecast. The forecasts may used to operate the ESS so as to achieve set objectives such as peak shaving and load-levelling by selecting periods which are expected to be off-peak periods and charging the ESS in those periods thereby increasing demand and then using energy from the ESS rather than importing energy from the electricity grid in the periods that are expected to be peak periods.

In general demand forecast aggregates for several customers are more accurate and it is harder to accurately predict the demand for a single customer [14]. Forecasts may be accurate or may have errors as shown in Figure 2.10 such as error in peak time, error in peak magnitude and error in peak duration.

Forecasts are generated using statistical methods such as linear regression and the quality of the schedule depends on the accuracy of the forecast. The forecasting methods and techniques are out of the scope of this study and all the analysis is based a historical average or historical record which is assumed to be the outcome of a forecast.

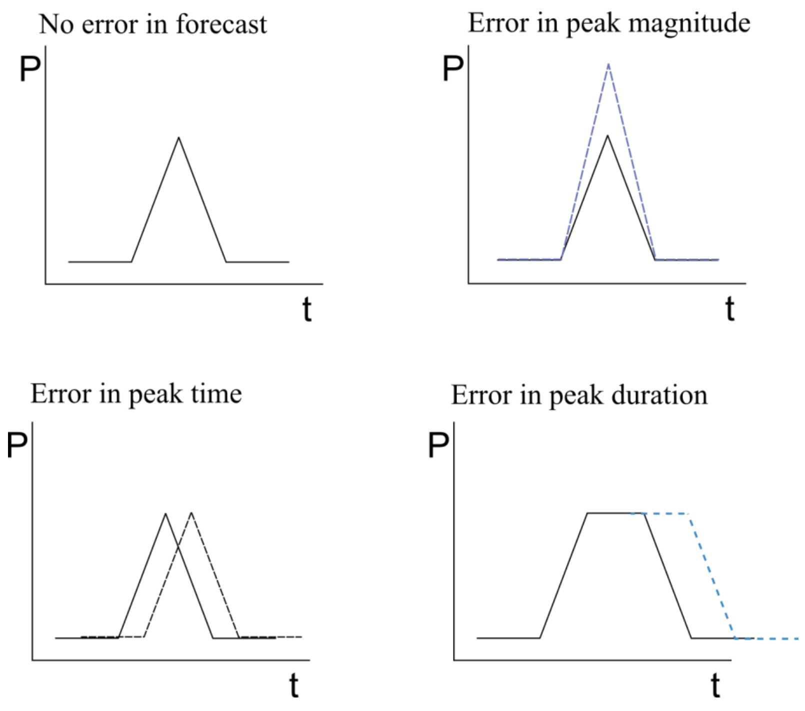


Figure 2.10. Possible outcomes of electricity demand forecasts

2.5 Review of Scheduling methods and algorithms

There are several methods currently in use for scheduling ESS operation in electrical systems for Active network management. Some of the most frequently used methods may be categorized into four categories according to their control schemes as Set-point control methods, Optimal Power Flow methods, Dynamic Programming and Artificial Intelligence methods, and Demand Response methods. These methods are reviewed in this section including their general procedure, strengths, and weaknesses.

2.5.1 Set-point control

Set-point control (SPC) involves using a reference value of demand to generate a schedule to charge or discharge the ESS. When the demand falls below the reference the ESS is charged and when it goes above the reference it is discharged, operating in a similar fashion to a thermostat or a closed loop control system as shown in Figure 2.11. The reference is based on a scale of the peak or an average calculated from aggregated demand values. On some occasions this average can be moved by a certain tolerance in response to different events. In some scenarios the set-point is chosen using forecasts a day ahead, and in some cases multiple set-points are used [14].

SPC is the standard control technique used on LV networks for controlling ESS. In [14] Rowe et. al describe how SPC is used on an LV network with an aggregated demand profile of 25 smart meters. In Figure 2.12 a 20kWh device with a discharge rate (rated power) of 4kW per half hour is added to the network. When the set-point is 12.5kW (i.e. the ESS charges when demand is below this value and discharges when it is above this value) the device is able to reduce the demand to the set-point for the entire duration of the data, resulting in demand reduction of 20%. When the set-point is 11.5kW the storage device runs out of energy before the end of the cycle resulting in a 7% demand reduction. This example illustrates that SPC can be effective when a correct set-point is selected and when a wrong set-point is used the same ESS capacity may not be as effective.

In [35] Papic uses SPC for a battery ESS at a TAB Mezica plant in Slovenia. TAB Mezica are a battery manufacturing company and decided to install a pilot device to showcase the use of ESS in load-levelling and peak demand reduction to avoid uprating a supply transformer. The method used here consists of three parts; the first part forecasts consumption in 15 minute intervals. If the consumption is found to exceed a desired maximum reference the BESS supplies power to the network, otherwise the ESS is put in charging mode. The reference power in this algorithm is set using the peak daily tariff for a certain day. During the peak daily tariff the reference value may be increased by 15%. A battery model that uses the voltage dependency of the battery on current and capacity is also used to check if the reference is suitable after the forecast has been determined. If the reference causes operation of the BESS outside tolerable values the demand from the battery is reduced. The paper concluded that based on the simulation results the reference value is significant for successful shaving of peak loads, therefore a poor choice will also result in ineffective peak reduction.

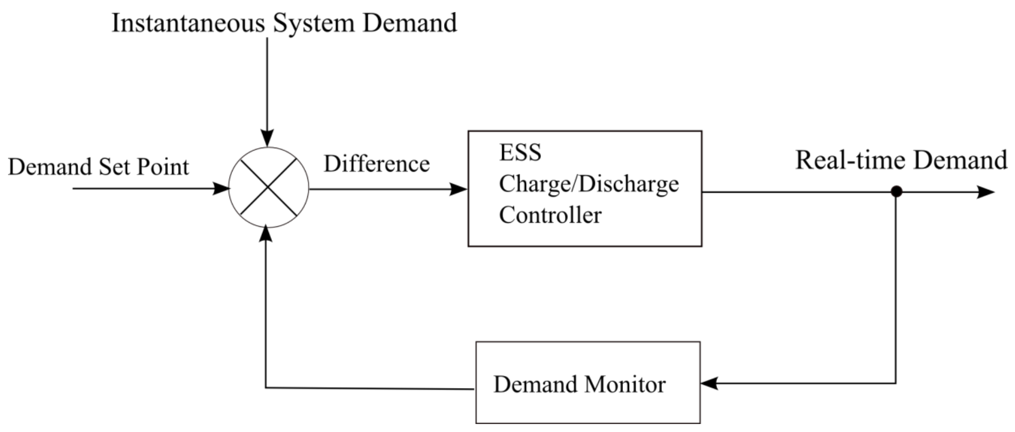


Figure 2.11. Simple Closed-loop system for set-point control in real-time ESS scheduling

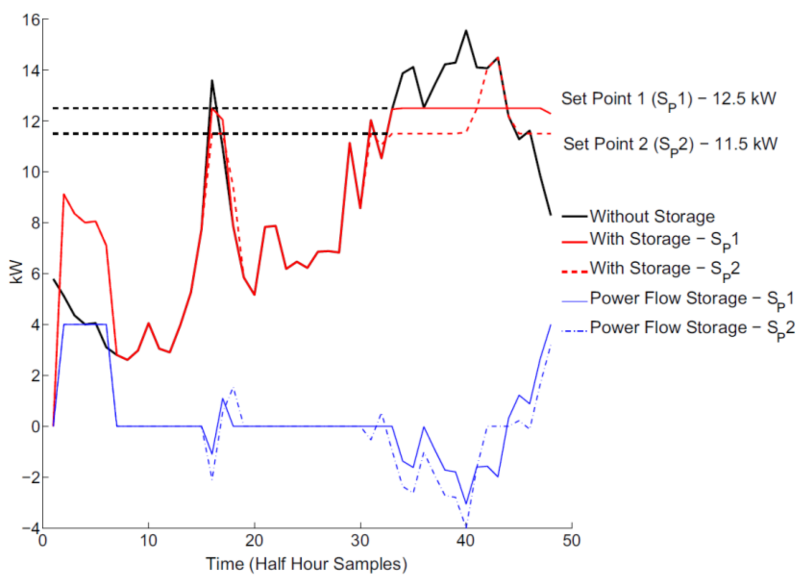


Figure 2.12. Set-point control example reducing demand over a 24 hour period using two different set-points

SPC may also be used by the consumer to reduce electricity demand at the grid level in return for incentives such as lower energy bills. In general most of the savings that are made by a consumer in a peak tariff-driven regime are made from reduction of the peak demand rather than in using electricity when it is cheaper [35].

In [36] Purvins *et. al* describe the application of battery-based ESS in household demand smoothening in electricity grids. Demand smoothening is another term for load-levelling and they've described it as a process by which the daily demand variations are reduced, achieved by charging a battery during valley demand and discharging it during peak demand. The study describes a time-dependent management model for a simple battery system, and a demand-tracking model for a more complex battery system. Both models implement SPC based on historic demand information.

The reference value for the SPC is calculated using the flowchart in Figure 2.13 which takes as input the hourly demand profile of the household, the BESS efficiency and a calculation step. The reference is first set using an average of demand throughout the day and then it is increased by the step value until the ratio of the amount of energy being supplied from the ESS to the amount being consumed by the ESS for charging is equal to the BESS efficiency. After the reference is obtained it is used to determine if the battery is to be charged or discharged in both models. This process is used to obtain a schedule for operating the ESS at a later time as part of an ANM scheme. The demand tracking model follows the instantaneous demand by further constraining the amount of charge or discharge according to the reference values, i.e. it does not charge or discharge to the maximum rated power but checks to ensure the system demand created by charging does not exceed the reference value and that the demand as a result of discharging is above the reference value. This enables it to provide a more level profile around the reference value.

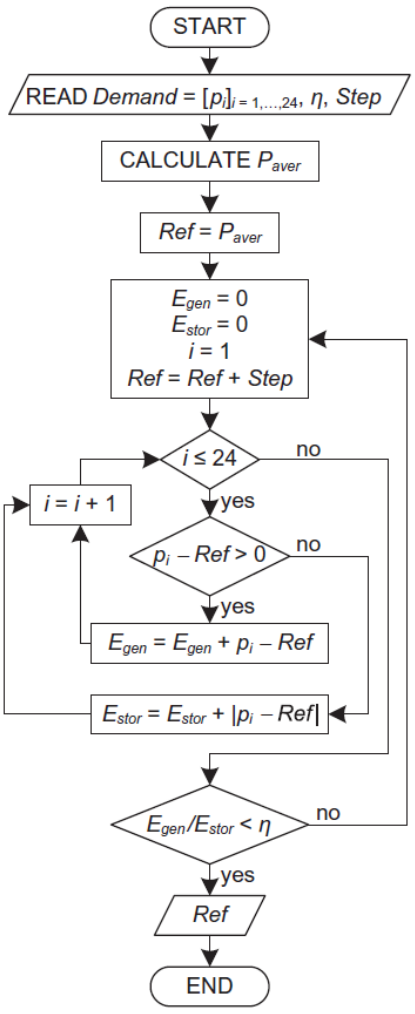


Figure 2.13. Flowchart for calculating set-point reference value for demand smoothening algorithm

SPC has an advantage of being the only real-time control option available to ESS control that does not have to depend on forecasts. It has an advantage of simplicity and ease of application to real time control at both the consumer level and the network operator level.

One of the disadvantages of this method is that it can be affected by short spikes in demand which will greatly alter references based on averages and therefore reduce the effectiveness. For example a “TV-pickup” which happens in the UK daily after popular television series end and consumers turn their kettles on around the same time [37]. SPC also sometimes operates without any knowledge of the future when used in real time control, therefore storage cannot be freed up for other support functions on the network[14]. Set-point control also requires real-time monitoring of the network, which is expensive [14].

Rowe *et. al* in [38] formulate a method using aggregated forecasts and an iterative scheduling algorithm. The method is based on operating a moving set-point which is determined for each iteration of the optimisation, hence it is a dynamic programming technique based on a moving set-point. The architecture as shown in Figure 2.14 combines both an off-line and on-line model, in which set-points are selected based on forecasts and the storage control system online enforces limits during real-time operation.

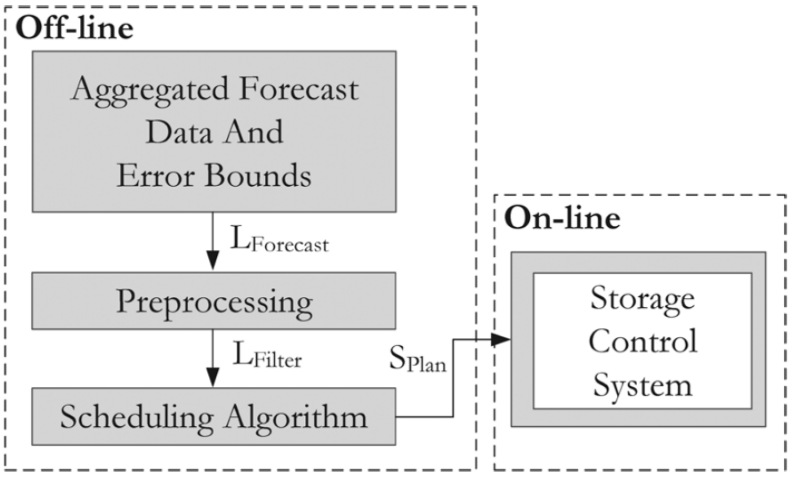


Figure 2.14. Architecture for off-line and on-line for scheduling based on aggregated forecast

2.5.2 Dynamic Optimal Power Flow

In power systems studies a power flow problem is related to finding the amount of power flowing through equipment such as lines and transformers in an interconnected network and the voltages at each node of the network. The theory of this problem and its solutions is covered in detail in chapter 4 of this thesis. In a practical power system all generators are not located at the same distance from load centres and their fuel costs are different. There is also usually more generation capacity than the total demand and power losses in a network and in an interconnected network there is a challenge of scheduling each power plant to minimise fuel cost.

Optimal power flow (OPF) is used to optimise the power flow problem of a large scale power system by setting limits within which the real and reactive power of generators may vary so as to meet a particular load demand with minimum fuel cost [3]. This is done by minimising selected objective functions while maintaining an acceptable system performance. The objective functions, or *cost functions,* may present economic costs, system security or other objectives. The functions are optimised using constrained parameter optimisation methods such as the Lagrange multiplier method with equality and inequality constraints imposed. In terms of the economic dispatch of a thermal power plant the fuel cost of a generator may be represented by a quadratic function of real power generation, which may be plotted as a fuel-cost curve. This cost curve is obtained by converting the ordinate of a heat-rate curve (in Btu/h) to a financial unit per hour (£/h) as shown in Figure 2.15. The cost function must be minimised and in so doing the operating power limits of the generator will be set.

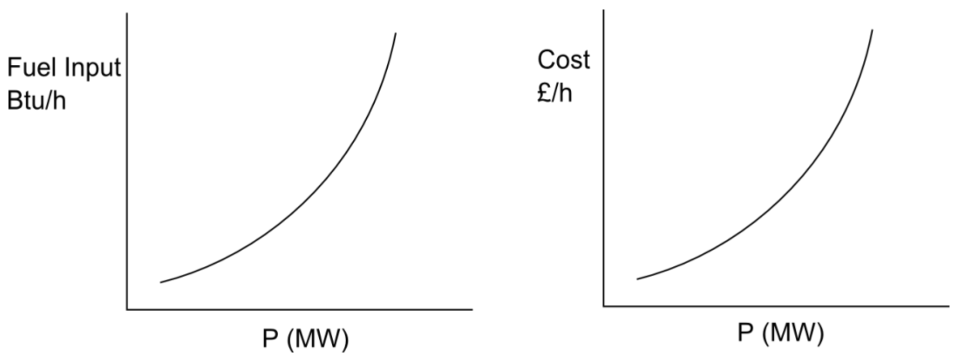


Figure 2.15. Heat rate curve and fuel cost curve used for generator cost function in optimal power flow problem

The problem is described mathematically as minimising the cost function (2.1):

$$f(x(t), y(t), z(t)) \quad \textbf{(2.1)}$$

subject to the equality constraints in (2.2):

$$g_i(x(t), y(t), z(t)) = 0 \quad i = 1,2, \dots, k \quad \textbf{(2.2)}$$

and the equality constraints in (2.3):

$$u_i(x(t), y(t), z(t)) \leq 0 \quad i = 1,2, \dots, k \quad \textbf{(2.3)}$$

where x is a control variable, y is a fixed variable and z is a derivative variable

In OPF the control variables (x) are adjusted by the optimisation process – the power output of a generator for example; the fixed parameters (y) such as thermal limits of lines define limits and the derived variables (z) are functions of the control variables and fixed parameters.

The concept of OPF is applied to ESS dispatch and scheduling in a number of research works. The objective function when integrating ESS may represent an operational objective to reduce power import at a point in a network where a sub-network is connected which known as grid connection point (GCP), to maximise export at a GCP from a generator in a sub-network, to increase penetration of renewables or to minimise energy costs.

A Dynamic Optimal power flow (DOPF) method is developed in [12] for using storage to maximise export and revenue. The same method is used in [29] for increasing the penetration of wind energy sources in the Shetlands UK and manage congestion in a distribution grid using ANM. This method extends the classical OPF formulation to cover multiple time periods by adding a parameter for inter-temporal variables and performing an OPF several times over a time period. Using DOPF the time horizon is broken up into multiple time steps and an OPF is performed at each time step to derive variables to control operation of the network.

In DOPF (2.1) – (2.3) will now include an inter-temporal variable, τ. For integrating ESS the inter-temporal variable may be the State of Charge (SOC) which is a fraction of the total energy capacity currently used dependent on the previous charge and discharge amounts [12]. The change in the SOC of the ESS is derived as a result of the optimisation process and keeps track of stored energy.

The ESS are modelled as generators that can inject positive or negative power into the network as shown in Figure 2.16, with the State-of-Charge as a discontinuous function related to the injections as follows, with P_{ESS} negative when charging and positive when discharging in (2.4):

$$\Delta SOC = \frac{-\Delta t}{E_{ESS}^{cap}} \begin{cases} \varepsilon_{in} P_{ESS} \,; P_{ESS} \geq 0 \\ \frac{1}{\varepsilon_{out}} P_{ESS} \,; P_{ESS} < 0 \end{cases} \tag{2.4}$$

The discontinuity cannot be accommodated in a nonlinear programming solution and is therefore removed by modelling the ESS as two separate generators for charging and discharging so that the ESS power is given as (2.5):

$$P_{ESS} = P_{ESS}^{charge} + P_{ESS}^{discharge} \tag{2.5}$$

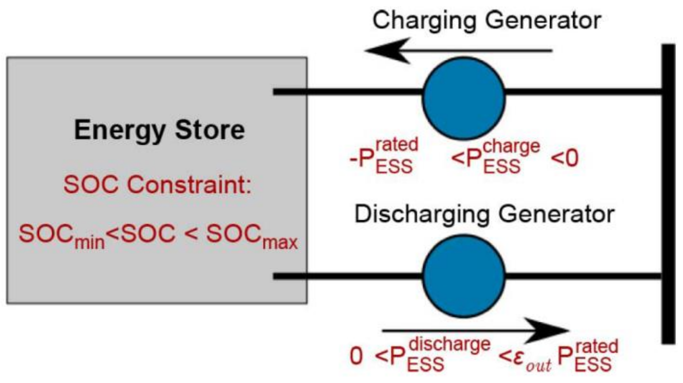

Charging Generator
Energy Store
SOC Constraint:
$SOC_{min} < SOC < SOC_{max}$
$-P_{ESS}^{rated} < P_{ESS}^{charge} < 0$
Discharging Generator
$0 < P_{ESS}^{discharge} < \varepsilon_{out} P_{ESS}^{rated}$

Figure 2.16. Two generator model of ESS

The State-of-Charge of the system at any given time is then expressed as the sum of all the changes in SOC due to the charging and discharging generators from prior time periods as (2.6):

$$SOC_{ESS}(t) = SOC_{ESS}(0) - \frac{\varepsilon_{in}\Delta t}{E_{ESS}^{cap}} \sum_{t'=1}^{t} P_{ESS}^{charge}(t') - \frac{\Delta t}{E_{ESS}^{cap}\varepsilon_{out}} \sum_{t'=1}^{t} P_{ESS}^{dis}(t') \tag{2.6}$$

The optimal change in SOC in (2.6) is achieved by deriving the amount of power P_{ESS}^{charge} that must be generated elsewhere in the network or imported from the GCP. If the cost of this power is positive in the objective function then minimising the value of $|P_{ESS}^{charge}|$ will lead to the optimal SOC. In this manner the charging and discharging of ESS may be scheduled to achieve the objectives set for the network.

An Active-Reactive Optimal Power flow is also formulated by Gabash and Li in [17] which uses wind forecasts and an electricity price model to determine the best periods to dispatch storage and maximise profit and reduce constraints on wind generation integration. In this method the charging and discharging periods are pre-defined according to the peak and off-peak pricing of electricity as shown in Figure 2.17. During periods T_1 and T_3 the demand, P_d, is below the wind power generated P_w, and the excess wind power is stored using ESS. This energy is released during the higher tariff period T_2. The time-based objective function is set to include the battery power, constrained by time periods for charging and discharging. The amount of power absorbed or injected by the BESS is determined by the amount which will maximise profits and reduce costs of energy losses. A similar method is used in [40] by Gabash and Li with a two-stage framework shown in Figure 2.18 where the upper stage is optimised using a Genetic Algorithm to determine the best period lengths for charging and discharging by selecting different combinations of time exhaustively before carrying out the OPF.

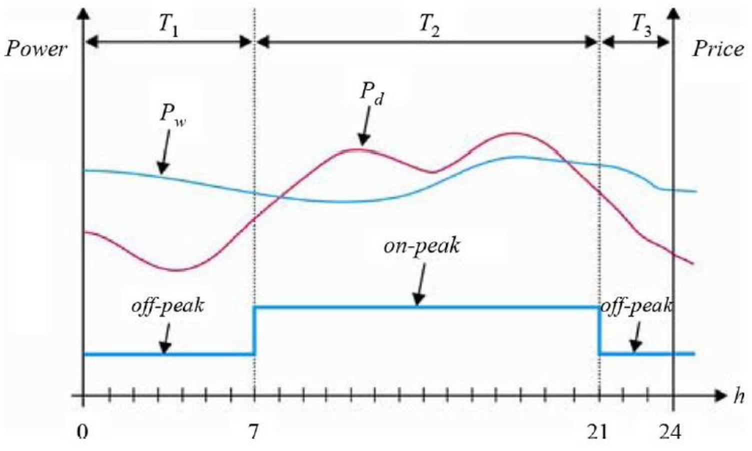


Figure 2.17. Active-Reactive Optimal Power Flow using electricity prices and wind power generation.

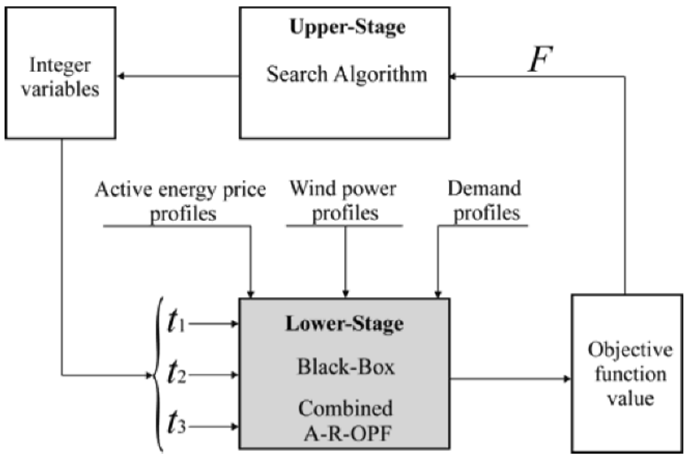


Figure 2.18. Active-Reactive Optimal Power Flow with search algorithm to determine best charging and discharging periods in an ESS

OPF methods are usually detailed and have several decades of research to rely on but they have a disadvantage of being computationally complex because of the addition of more variables to an already complex non-linear problem. Optimal Power Flow based methods also rely heavily on generation price information or other forecasts which are not always available or in the control of the distribution network operator or a customer.

2.5.3 Dynamic Programming and Artificial Intelligence methods

Dynamic programming (DP) is a technique for efficiently implementing a recursive algorithm by storing partial results [41]. A recursive algorithm is a method that solves a problem by calling a function several times but each time using a smaller value as an argument until the problem can be solved directly. A simple example is finding the factorial of a number by gradually finding the factorial of smaller numbers. When a naïve recursive algorithm computes the same sub-problems repeatedly the solutions may be stored in a lookup table to improve the efficiency of successive solutions. They are effective in optimisation problems on combinatorial objects that have an inherent left to right order among components, such as incremental time-based scheduling [41].

Applied to ESS, Qin *et. al* in [42] use a quadratic fuel cost curve and recursively solve finite space Markov decision processes for each sub-network under consideration with AC power flow equations as part of a Dynamic Programming model. The energy stored in a device at the *i*th bus is shown to be computed as a function of the model from the preceding time-step. If the energy storage is co-located with a load, they first supply the load locally with energy storage and more energy is then drawn from the grid if it is necessary to meet the demand. In particular, we model the dynamics of stored energy in the idealized storage (no losses) in (2.7):

$$x_i(t+1) = x_i(t) + u_i(t) - d_i(t) \tag{2.7}$$

Where $x_i(t)$ is the energy stored at the *i*th bus at interval *t*, $u_i(t)$ is the power drawn at the bus and $d_i(t)$ is the demand at the bus.

Sioshansi *et. al* in [43] assume knowledge of future energy prices and perform an optimal dispatch based on this and then assign probability distribution for the availability of storage based for each successive periods in the DP. In [44] Neves *et. al* use price based demand response and DP to determine the best transition state from one time step to the next via an economic dispatch which is based on Quadratic programming. In [45] DP and Optimal Power flow are combined to dispatch energy storage for load levelling and peak shaving at grid level.

Dynamic programming (DP) methods while being detailed and quite effective in handling nonlinearity and stochastic operation also depend on a specific value function which is based on generation data or cost data for each time step and are usually also computationally complex. If the specific value function is not defined properly the results will have poor quality.

Artificial intelligence (AI) is also being applied to ESS scheduling mostly in forecasting demand before applying another method for scheduling. Artificial Neural Networks for example are used to forecast household demand a day ahead in [46] based on the historic data from three months prior to the day. Based on the forecast and solar energy photovoltaic generation a schedule is generated for operating energy storage to trade in an energy market after local demands are satisfied in a manner similar to set-point control.

2.5.4 Demand Response and tariff-based Methods

Demand response(DR), or Demand Side Response (DSR), is defined as changes in electric usage by demand-side resources from their normal consumption patterns in response to changes in the price of electricity over time or to incentive payments designed to induce lower electricity use at times of high wholesale market prices or when system reliability is jeopardized [47].

DR is one of the interesting features of future electricity networks and the smart electricity grid and is a popular topic in recent research work. The concept of a smart grid is essentially the current electricity grid with 2-way communication infrastructure such that consumers may receive information from suppliers and supplier may receive consumption information more efficiently.

When customers participate in DR there are three possible ways in which they can change their use of electricity [48]:

i. Reduce energy consumption through demand curtailment strategies

ii. Moving energy consumption to a different time

iii. Using on-site standby generated energy to reduce their dependence on the grid

For the application of ESS in DR moving energy consumption to a different time is the option that is used. Customers comfort may be limited by DR and therefore automation, monitoring and control technologies are fundamental to making DR less of a hindrance to customers [49]. These automation technologies include “smart” devices which can respond to changes in electricity price and deferrable loads. These include electric vehicles (EV), refrigerators, dishwashers, washing machines, and energy storage devices. The effect of peak shaving and load-levelling is achieved if the correct price signals are sent to these devices during peak periods to encourage them to cut off or reduce their electricity consumption.

The report by Strbac *et. al.* on demand response based management of distribution network highlights the benefits of smart charging of EVs as compared to business-as-usual approach [2]. Figure 2.19 shows the changes in demand profile for the case in a residential area driven by the EV charging when people return home from work for both business-as-usual and smart modes of operation. In the normal mode all EVs charge as soon as they are plugged in during the evening hours and create a rise in the peak demand, but in the smart approach the charging is spread out to the early morning hours when most customers are asleep.

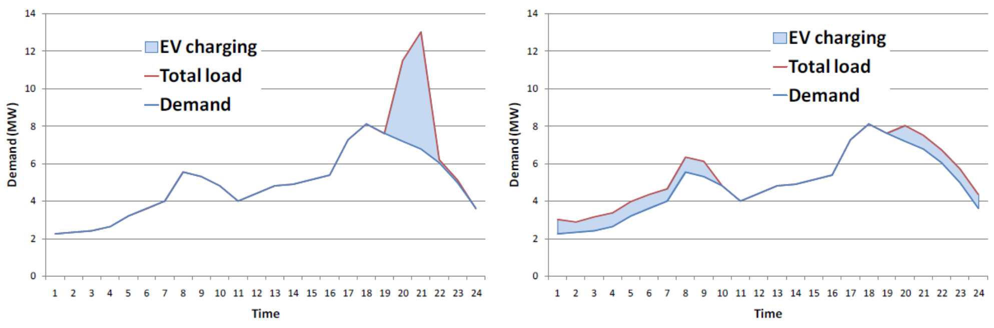


Figure 2.19. Business-as-usual (left) and Smart (right) charging in a residential area (8,000 properties) driven by charging of 5,000 EVs when people return from work

In [47], EVs and smart appliances are used at the consumer level to shape the demand on the network by responding to price signals by prioritising the appliances according to customer preference. The ESS in this case are the EVs and the energy stored is used for mobility.

In some cases the stored energy is returned to the household or the sold on the grid in a Vehicle-to-home (V2H) or Vehicle-to-grid (V2G) setup. In [50] Zhao et. al. describe a peak shaving strategy that charges an EV during off-peak periods and once a certain state-of-charge is attained a V2H program discharges the electricity to the home – which is similar to a set-point control method since that SOC is determined beforehand and not dynamic. They also set constraints on the amount of demand that can be handled by the EV.

Heymans et. al. in [51] describe the application of second-use EV batteries in households for energy storage and peak shaving. Their ESS scheduling approach combines demand information and financial incentives to minimise the costs of energy using Time-of-use pricing. In some cases the peak demand increases or shifts to a new period, which highlights a possible impact of naïve scheduling based on only financial incentives.

The DR scheme in [9] requires control of "wet appliances" to reduce congestion by scheduling the time they use energy to coincide with low demand periods which customers may find intrusive.

Demand response (DR) schemes based on Time of Use (ToU) pricing may lead to peak demand reduction but are generally out of the control of the Distribution Network Operator and consumer. DR schemes depend on consumer behaviour in response to energy prices but the DNO in most cases do not set the energy prices, the energy supplier does. The consumer may also not respond to the prices in such a manner that it leads to peak demand reduction.

3 Application of Combinatorial optimisation methods to Energy Storage System Scheduling

3.1 Introduction

The scheduling of ESS operation in an electrical system is formulated in this section as an optimisation problem of maximising the utilisation of assets within the operational constraints of a given electrical network or electrical system such as loading, thermal, rated power, and energy capacity to postpone infrastructure reinforcement and provide other benefits.

Firstly, the problem is modelled as a form of the bin packing problem and the subset sum problem. These are part of a class of problems known as *combinatorial optimisation* problems, which are described as a set of problems that consist of finding an optimal object from a finite set of objects where an exhaustive search is not feasible i.e. one cannot feasibly test all possible inputs to determine which input produces the optimal output in reasonable time. In these problems the set of feasible solutions are discrete or can be reduced to discrete and from which the best solution is found [41][52].

After the problem is defined clearly a methodology is presented for finding an optimal solution using simple heuristic methods for solving both problems with slight modifications. Heuristic methods apply a practical, “rule-of-thumb” approach to speed up searching through solutions. The methodology is then extended by applying Genetic Algorithm (GA) optimisation.

3.2 Problem Definition

The problem of operating a network within thermal and capacity constraints using ESS is defined as a dynamic optimisation problem of selecting the best periods to charge and discharge the ESS to keep an otherwise overloaded network within operational limits. This is achieved by a peak shaving and/or load-levelling objective.

Transforming a continuous daily demand profile into b discrete time steps with normalized demand D_i in the ith interval, the objective is to minimise the peak demand given as D_p in (3.1) within the time horizon $t_1, \dots, t_b$. The integer b will be 24 for an hourly resolution and 48 for half hourly, etc. (3.2) is a peak shaving objective.

$$D_p = \max\left(\{D_1, \dots, D_b\}\right) \tag{3.1}$$

$$F_1 = \min(D_p) \tag{3.2}$$

The load-levelling is defined as the difference between the peak and trough demand, which gives an indication of how “flat” the demand profile is. It is given in (3.3) and the objective to minimise the variation in demand is given in (3.4).

$$D_L = \max(\{D_1, \dots, D_b\}) - \min(\{D_1, \dots, D_b\}) \tag{3.3}$$

$$F_2 = \min(D_L) \tag{3.4}$$

When combined with ESS utilisation F_1 and F_2 are constrained by ESS total energy storage capacity E, ESS rated power P, efficiency μ, and network capacity C. If the amount of energy stored or used during the *i*th interval is E_i the constraints are as follows:

ESS Capacity Constraint

The amount of energy stored or released during a time period *i* depends on its operation at a previous time interval i.e. it cannot discharge energy that was not stored previously or store energy beyond its available capacity, given in (3.5):

$$\left|\sum_{i=1}^{b} E_i\right| < E \tag{3.5}$$

where $\{-E < E_i < 0 \; or \; 0 < E_i < E\}$ for discharging and charging phases respectively

ESS rated power constraint

The demand met by the ESS discharge is constrained to its operating power, given in (3.6):

$$\frac{E_i}{t_i} \leq P \tag{3.6}$$

for all *i*.

Network capacity constraint

Total energy demand in the network or system must not exceed the firm capacity, given in (3.7):

$$C = \sum_{i=1}^{b} D_p t_i \tag{3.7}$$

ESS charging must not violate the loading and thermal constraints of the network assets injecting energy at the GCP, given in (3.8):

$$\sum_{i=1}^{b} D_i t_i + \mu E_i \leq C \tag{3.8}$$

Equations (3.5) – (3.8) define the constraints for the optimisation. The solution will be an ordered set representing the ESS dispatch schedule $\{E_1, E_2, \dots, E_b\}$ and the final demand profile $\{D_1, D_2, \dots, D_b\}$. The State-of-Charge is resolved by running both cycles and merging the results.

As a result of the initial discretization of the demand profile, the ESS operation is also implicitly discretized. However the ESS dispatch schedule allocation is not bound by the same discretization criteria i.e. any E_i may be represented as $E_i = \{E_{i1}, E_{i2}, \dots\}$.

Essentially, the cardinality of the set representing ESS dispatch schedule is not restricted to the cardinality of the set representing the demand profile. For example an hourly resolution with 24 possible charge or discharge points may have an infinite number of charge/discharge amounts in 24 groups.

The search space for the solution is defined in the possible elements of the storage allocation set which is infinite without discretization. However various solutions still exist in the finite set which is flexible.

A suitable solution will have the following characteristics:

i. The solution must globally and recursively target the points in the demand profile where the most violations or highest peaks occur and solve those first and then find the next point closest to a constraint violation
ii. ESS must not be charged at points that will create new peaks which were operating within constraints previously
iii. Operation of the ESS should not increase the energy demand in the network as efficient operation will involve a full charge and a full discharge

Some solutions will meet the set objectives more optimally than others and in cases where a global optimum cannot be guaranteed the search may be terminated when a sufficiently suitable solution is found.

The method proposed for obtaining a solution is a multi-stage optimisation heuristic involving first generating a set of different solutions which may be optimal or suboptimal and then further optimising by combining the best characteristics from the set to improve on the best solution.

3.3 Allocating ESS capacity from a search space using a subset sum algorithm

As indicated in the problem definition, the ESS capacity may be defined as the sum of a set of various combinations which is essentially a form of the subset sum problem. The subset sum problem is a problem of finding a subset that adds up to a particular value from a larger set [53]. It is a form of the knapsack problem, and it arises in situations where a quantitative target should be reached such that its negative deviation (such as loss of space, time, money, etc.) must be minimised and a positive deviation is not allowed.

If the list contains a small number of integers, then an exhaustive search can be used to solve the problem. For larger sets or real numbers other techniques such as dynamic programming and backtracking algorithms may be applied to solve it exactly. It may be expressed as a {0,1} integer relation problem as follows:

Let $K = \{k_1, k_2, k_3 \ldots . k_k\}$ be a finite set and m be a positive integer called the target. Associated with each $k_i \in K$ is a positive integer $s(k_i)$ called its size. Does there exist a solution to (3.9)?

$$\sum_{i=1}^{k} s(k_i) x_i = m \qquad \textbf{(3.9)}$$

where $x_i = 0$ or $x_i = 1$?

In this methodology the search space for finding that target sum is created from a set of random numbers to be filtered using a subset sum algorithm as shown Figure 3.1. The search space is bounded by a predefined number of elements to make up the solution of the subset sum algorithm. Using this method the ESS capacity may be allocated in various combinations of smaller units.

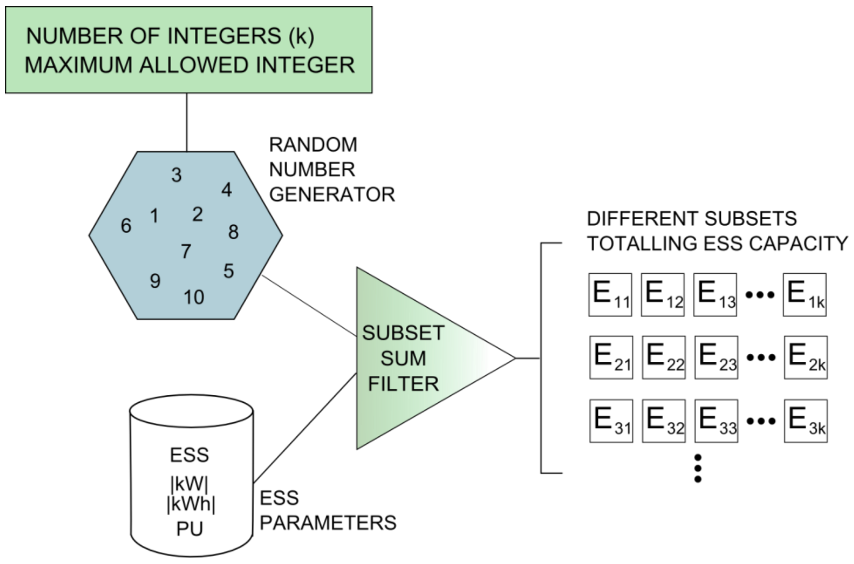

NUMBER OF INTEGERS (k)
MAXIMUM ALLOWED INTEGER
RANDOM NUMBER GENERATOR
3 4 6 1 2 8 7 9 5 10
SUBSET SUM FILTER
ESS |kW| |kWh| PU
ESS PARAMETERS
DIFFERENT SUBSETS TOTALLING ESS CAPACITY
E_{11} E_{12} E_{13} ••• E_{1k}
E_{21} E_{22} E_{23} ••• E_{2k}
E_{31} E_{32} E_{33} ••• E_{3k}

Figure 3.1. Allocation of ESS capacity from subset sum filter and random number generator

Backtracking method is applied to solve the SSP by searching for a solution within a larger set of numbers with a binary coefficient indicating if a number exists in the solution. Backtracking is a systematic way to iterate through all the possible configurations of a search space. These configurations may represent all possible arrangements of objects (permutations) or all possible ways of building a collection of them (subsets) [41]. In this case a decision tree is used to generate all possible solutions.

A novel approach is applied to the original SSP by modifying it to include a tolerance variable. This is shown in the decision tree in Figure 3.2, where the augmentation threshold *t* represents the number of integers at which point the sum is made up by an adequate addition of one or more integers to the set before the maximum cardinality *k* is reached. For any $x_i = 0$ before the augmentation threshold, which is the number of bits that defines a close enough solution, the path is eliminated. When the bit tolerance level is reached at the augmentation threshold the difference between the target sum and the running sum, ***a***, is added to the current set to obtain the solution.

From the binary tree in Figure 3.2 it can be seen that only the solutions along the left branches can be active in the solution. This is because the elements in the set are sorted before the search algorithm begins and because of the second condition for continuation.

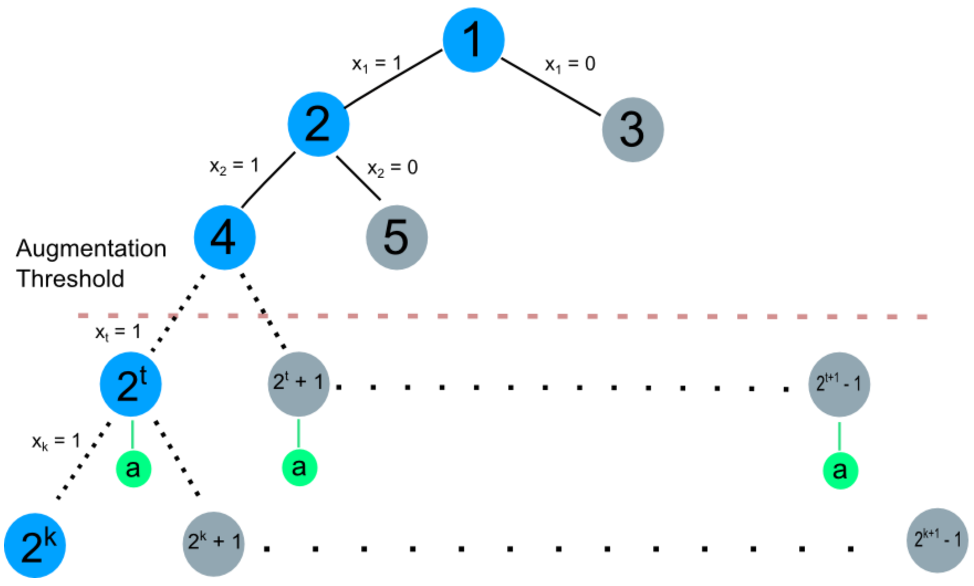


Figure 3.2. Decision tree for subset sum generation with augmentation threshold

Due to the heuristic and randomized method of generating subset sums, a given maximum number of blocks may yield different solutions. For example a 20kWh ESS may yield different combinations as shown in Figure 3.3.

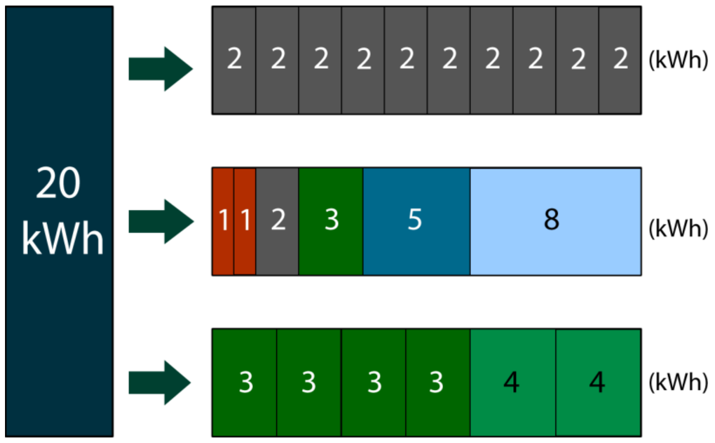


Figure 3.3. Possible combinations of blocks for a 20kWh ESS

The flowchart of the subset sum generator is shown in Figure 3.4. The algorithm that the flowchart follows is explained in the following steps:

1. Choose as inputs the ESS capacity, **E**, and a tolerance, **ε**, for the cardinality of the solution set, **S** i.e. the number of numbers in the set
2. Randomly generate a set of **k** integers to choose a solution from. This set is named **K**
3. Initialise a running sum, **R**, to keep track of the numbers being summed to find the solution, and X a binary vector that has bits set to 1 if number is in running sum and 0 if

not in solution. Use variable **n** to indicate amount of numbers in solution and **r** for numbers in running sum

4. Check if ESS capacity **E** is greater than the sum of numbers in random set **K** and if so regenerate random set, if not sort random set in ascending order
5. Check if running sum **R** is less than or equal to ESS capacity **E.** If equal, then binary vector X corresponds to solution and end the algorithm. If less keep including numbers from **K** until tolerance margin (n-ε) is reached and then introduce a new number **a** to add to **R** such that it makes up solution and ends algorithm.
6. If running sum exceeds ESS capacity or tolerance isn't reached then return to step 2 and generate new random set and restart running sum.

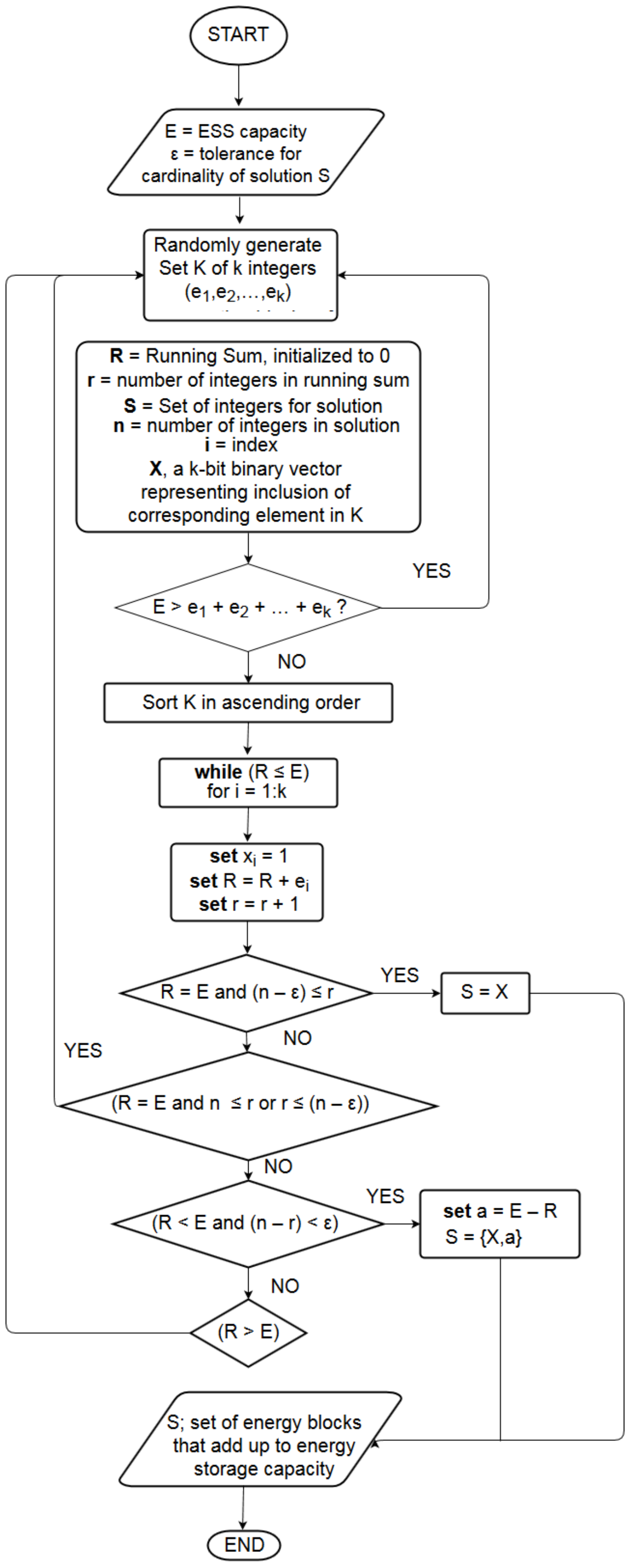

START
E = ESS capacity
ε = tolerance for cardinality of solution S
Randomly generate Set K of k integers (e_1,e_2,…,e_k)
R = Running Sum, initialized to 0
r = number of integers in running sum
S = Set of integers for solution
n = number of integers in solution
i = index
X, a k-bit binary vector representing inclusion of corresponding element in K
$E > e_1 + e_2 + … + e_k$?
YES
NO
Sort K in ascending order
while (R ≤ E) for i = 1:k
set x_i = 1 **set** R = R + e_i **set** r = r + 1
R = E and (n – ε) ≤ r
YES
S = X
NO
YES
(R = E and n ≤ r or r ≤ (n – ε))
NO
YES
(R < E and (n – r) < ε)
set a = E – R
S = {X,a}
NO
(R > E)
S; set of energy blocks that add up to energy storage capacity
END

Figure 3.4. Flowchart for solution to subset sum problem for ESS energy distribution framework

The pseudo-code for the algorithm is given as follows in listing 3.1

Listing 3.1. Pseudo-code for solution to Subset sum problem for ESS energy distribution

```
Let K = ($e_1,e_2,…,e_k$), a set K of k integers representing blocks of energy, generated randomly

Let X = ($x_1,x_2,…,x_k$), a k-bit binary vector representing the solution initialized to (0,0,…,0) with each bit representing the inclusion of the corresponding element in K

Let E be the ESS capacity such that $E \leq e_1 + e_2 + … + e_k$

Let S be a set that adds up to the target sum E

Let n be a number of integers chosen for a valid solution, representing the cardinality of S

Let $\varepsilon$ be the tolerance for the minimum allowed cardinality of S

Let R be a running sum of consecutive integers initialized to 0

Let r be the number of integers added up for the running sum

Let a be the augmentation integer or set used to correct a close enough solution

Let i be an index variable

if ( $E > e_1 + e_2 + … + e_k$ )
    generate new random set K and restart algorithm
else
    sort K in ascending order
    while ($R \leq E$)
        for each i from 1 to k
            set $x_i$ = 1
            set R = R + $e_i$
            set r = r + 1

            if (R = E and $(n - p) \leq r \leq n$ )then
                return S = X as the solution //subset found
            else if (R = E and $n \leq r$ or $r \leq (n - \varepsilon)$) then
```

```
                    return false and restart algorithm // out of range
                else if (R < E and (n - r) < ε) then
                    set a = E - R
                    return S = {X,a} as the solution // augmented set
                else if (R > E)
                    return false and restart algorithm // too large
```

3.4 Scheduling ESS allocation in demand profile using the Bin Packing Problem and solutions

The strategy adopted for scheduling the ESS allocations in the demand profile is to discretize the demand profile into a number of intervals with normalized values for demand as illustrated in Figure 3.5 and then allocating the ESS units generated from the subset sum algorithm to the discrete intervals.

A histogram is created out of the demand profile by first *binning* a cycle into a set of *b* discrete consecutive time intervals of a given duration *t* (for example 1 hour intervals will result in 24 discrete “bins” over 24 hours, 15-minute intervals will result in 96 bins). The demand values in hourly resolutions are normalized in the time-step of one hour. Therefore shorter peaks will be accounted for if an average is used for normalization but will be missed if they occur in the same time step as a higher local peak and normalization in the time-step is done against maximum demand. A smaller resolution will increase accuracy in practice as the original profile is continuous and the algorithm discretizes it.

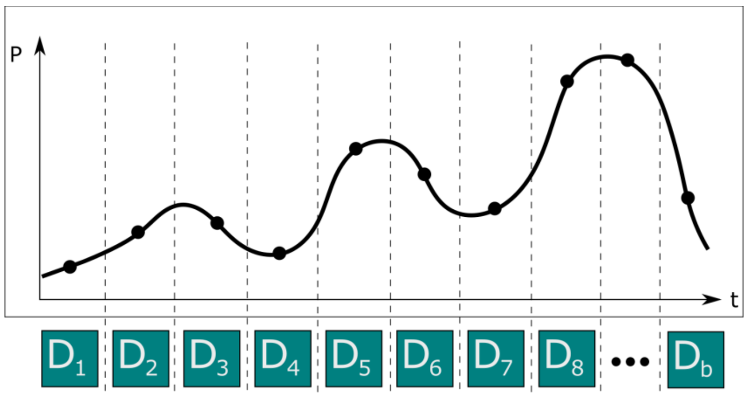


Figure 3.5. Normalization and discretization of demand profile into bins for the bin-packing algorithm

To satisfy the conditions set for a suitable solution the unit allocations must not lead to the current maximum demand being exceeded during charging, and during the discharge it must globally target peaks. This describes the allocation problem as a form of the bin-packing problem.

The bin-packing problem is a problem of packing a set of items into a number of bins such that the total weight, volume or some other parameters does not exceed some maximum value [54], [55]. Bin packing problems have many industrial applications including filling up shipping containers, loading trucks to a certain weight, storage of items in warehouses, storage of data on disk drives and machine scheduling problems involving several machines performing independent tasks. It is an appealing mathematical model, yet work on this problem as an organized topic is only about 35 years old, and therefore fairly recent [56].

Bin-packing problems are usually solved heuristically. The main heuristic methods of obtaining a solution are shown in Table 3.1 [55]. An efficient solution that involves first ordering items in order of size before placement is never sub-optimal by more than 22% [54].

Table 3.1. Heuristics for solving bin-packing problem

	Heuristic	Procedure
1	Next Fit (NF)	Place items in the current bin if it fits, if not close that bin and start a new bin
2	First Fit (FF)	Place items in the *lowest numbered bin it fits in*. If there is no such bin start a new bin
3	Best Fit (BF)	Place items in the bin that leaves the *least capacity left over after placement*. If there is no such bin start a new bin
4	Worst Fit (WF)	Place items in the bin that leaves the *most capacity left over after placement*. If there is no such bin start a new bin

The process of charging and discharging using a solution for the bin-packing problem is described in the next sub-sections. As indicated earlier, the items to be packed are the ESS energy blocks from the solution to the subset sum problem.

3.4.1 Charging ESS using solution of demand profile bin packing problem

After binning the demand profile (Figure 3.5) and the generation of ESS energy blocks from the solution of the subset problem as items to be packed (Figure 3.1 – Figure 3.4), the blocks are packed using the worst-fit heuristic solution. The worst-fit heuristic is chosen because it places items in the bin that leaves the most capacity left over after placement [55]. Therefore the WF will target the periods with the least demand as they will leave the most space over after placement of ESS demand and prevent charge in periods of highest demand thereby producing a load-levelling effect and demand time shifting (Figure 3.6).

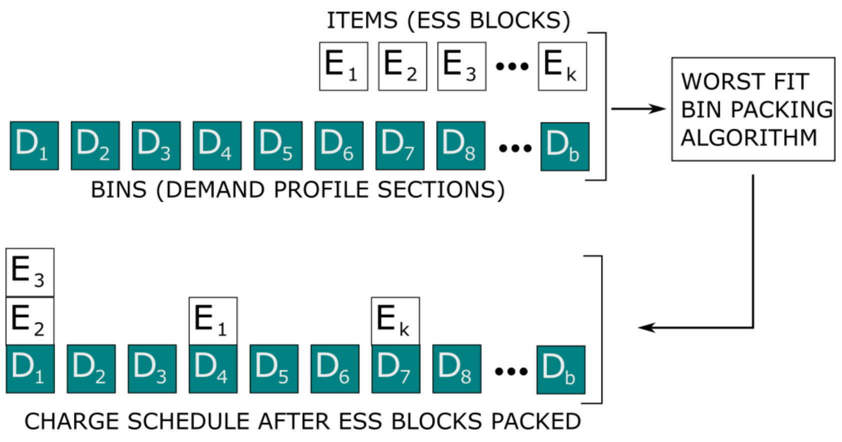

ITEMS (ESS BLOCKS)
E_1 E_2 E_3 ••• E_k
WORST FIT BIN PACKING ALGORITHM
D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8 ••• D_b
BINS (DEMAND PROFILE SECTIONS)
E_3
E_2
E_1
E_k
D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8 ••• D_b
CHARGE SCHEDULE AFTER ESS BLOCKS PACKED

Figure 3.6. Charge scheduling of ESS units using bin-packing algorithm

The flowchart for charging the ESS using a bin-packing heuristic is given in Figure 3.7 and the steps of the algorithm are as follows:

1. Set as inputs the binned demand profile as a set of bins **B** of capacity **V** with their already utilised capacity representing the demand in the time step, make an identical copy CB to represent charging profile.
2. Using solution of ESS capacity subset sum problem, a set of virtual energy blocks, as items to be packed into the bins sort the items in ascending order of energy magnitude. Make a set **M** to keep unused energy blocks
3. Sort bins CB in order of utilised capacity from lowest to highest and select emptiest bin
4. Place energy blocks in this bin until the next block cannot fit without exceeding capacity and then move to the next emptiest bin and fit this block until the energy blocks are used up
5. CB represents the new demand profile during the charging phase of the ESS and gives the optimal charging periods and amounts for load levelling.

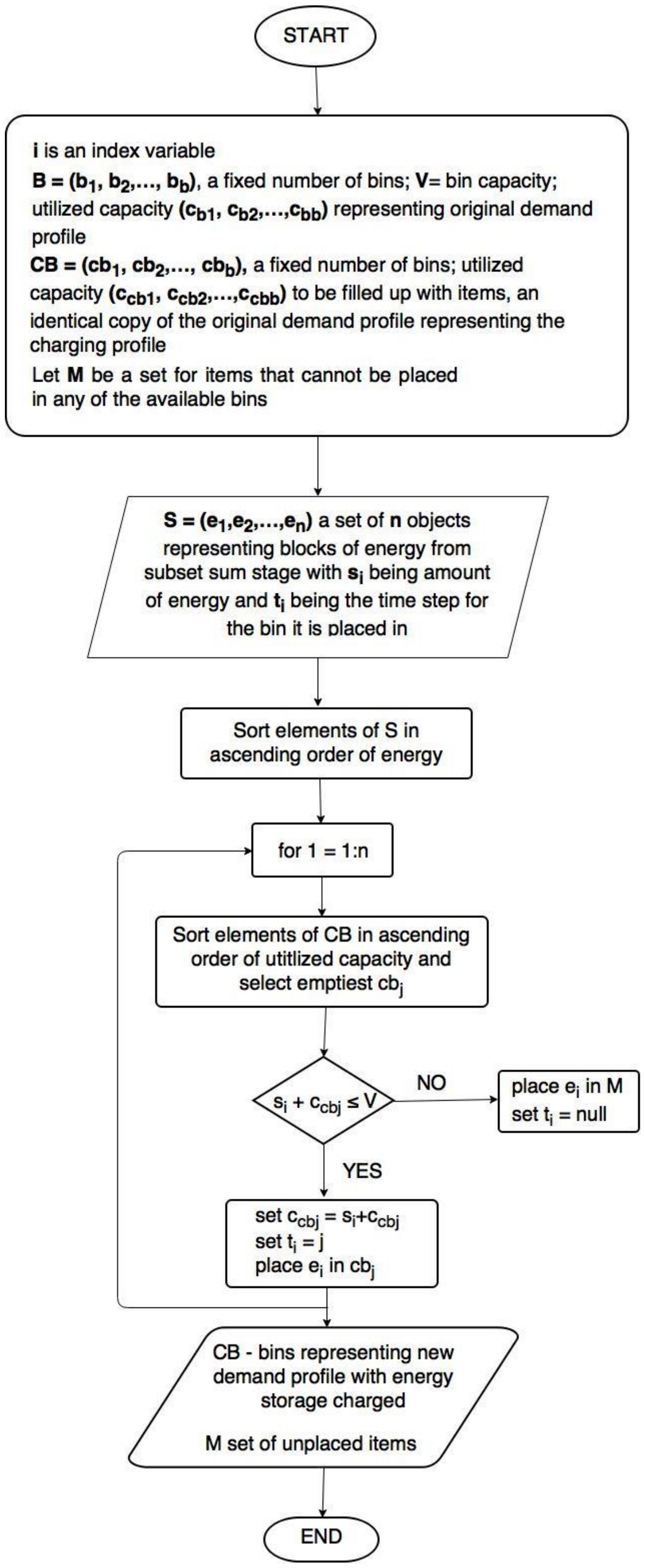

START
i is an index variable
B = (b$_1$, b$_2$,…, b$_b$), a fixed number of bins; **V**= bin capacity; utilized capacity **(c$_{b1}$, c$_{b2}$,…,c$_{bb}$)** representing original demand profile
CB = (cb$_1$, cb$_2$,…, cb$_b$), a fixed number of bins; utilized capacity **(c$_{cb1}$, c$_{cb2}$,…,c$_{cbb}$)** to be filled up with items, an identical copy of the original demand profile representing the charging profile
Let **M** be a set for items that cannot be placed in any of the available bins
S = (e$_1$,e$_2$,…,e$_n$) a set of **n** objects representing blocks of energy from subset sum stage with **s$_i$** being amount of energy and **t$_i$** being the time step for the bin it is placed in
Sort elements of S in ascending order of energy
for 1 = 1:n
Sort elements of CB in ascending order of utitlized capacity and select emptiest cb$_j$
$s_i + c_{cbj} \leq V$
NO
place e$_i$ in M
set t$_i$ = null
YES
set c$_{cbj}$ = s$_i$+c$_{cbj}$
set t$_i$ = j
place e$_i$ in cb$_j$
CB - bins representing new demand profile with energy storage charged
M set of unplaced items
END

Figure 3.7: Flowchart for charging ESS using bin-packing algorithm

The pseudo-code for the charging cycle is given in Listing 3.2.

Listing 3.2. Algorithm for charging ESS using solution of Demand Profile Bin Packing Problem

```
Let i be an index variable

Let S = ($e_1$,$e_2$,…,$e_n$), a set of n objects representing blocks of energy
with properties $s_i$ being the amount of energy and $t_i$ the charge time
step representing the bin it is placed into, to be packed as items
into a fixed number of bins

Let the size of each item be the amount of energy in the block

Let B = ($b_1$, $b_2$,…, $b_b$), a fixed number of bins with a fixed capacity V
and utilized capacity ($c_{b1}$, $c_{b2}$,…,$c_{bb}$) representing the original demand
profile

Let CB = ($cb_1$, $cb_2$,…, $cb_b$), a fixed number of bins with a fixed
capacity V and utilized capacity ($c_{cb1}$, $c_{cb2}$,…,$c_{cbb}$) to be filled up
with items, an identical copy of the original demand profile
representing the charging profile

Let M be a set for items that cannot be placed in any of the
available bins

Sort elements of S in ascending order of size

for each i from 1 to n:

    Sort elements of CB in ascending order of capacity utilized and
    select emptiest

    Let $cb_j$ be the selected emptiest bin
    if ($s_i + c_{cbj} \leq V$):
        set $c_{cbj} = s_i + c_{cbj}$
        set $t_i = j$
        place $e_i$ in $cb_j$
    else:
        place $e_i$ in M
        set $t_i$ = null
```

3.4.2 Discharging ESS as bin packing problem

The most important aspect of this stage is the transformation of the original demand profile. To formulate the discharge phase as a bin packing problem the original demand profile goes through a *horizontal mirror transformation*, i.e., the highest demand becomes the lowest demand and vice-versa. This is done to favour the highest demand time steps in the packing algorithm that is used to distribute the energy blocks that have been charged.

The worst-fit heuristic algorithm is also applied to pack the charged energy blocks the previously highest periods of demand which are now the lowest demand period as a result of the inversion. Alternatively the best fit method may be used on the original profile to place ESS units with negative values to create a discharge (Figure 3.8).

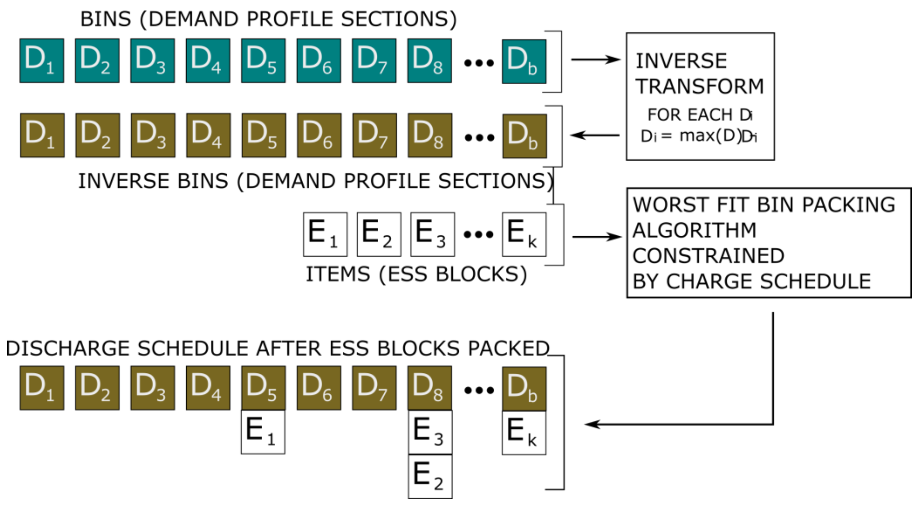

BINS (DEMAND PROFILE SECTIONS)
D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8 ••• D_b
INVERSE TRANSFORM
FOR EACH D_i
$D_i = \max(D) D_i$
INVERSE BINS (DEMAND PROFILE SECTIONS)
E_1 E_2 E_3 ••• E_k
ITEMS (ESS BLOCKS)
WORST FIT BIN PACKING ALGORITHM CONSTRAINED BY CHARGE SCHEDULE
DISCHARGE SCHEDULE AFTER ESS BLOCKS PACKED

Figure 3.8. Discharge scheduling of ESS units using bin-packing algorithm and inverse transform

The flowchart for discharging the ESS using a solution of a bin-packing problem is given in Figure 3.9.

The steps in the algorithm for Figure 3.9 are as follows:

1. Set as inputs the binned demand profile as a set of bins **B** of capacity **V** with their already utilised capacity representing the demand in the time step
2. Make a mirror transformation of the original profile by subtracting utilised capacity of each bin from the total capacity, producing an inversion effect as the lowest filled bin will now be the most filled and vice versa. Label this inverse copy as DCB to represent discharging profile.
3. Using solution of ESS capacity subset sum problem, a set of virtual energy blocks, as items to be packed into the bins sort the items in ascending order of energy magnitude. Make a set **DM** to keep unused energy blocks

4. Sort bins DCB in order of utilised capacity from lowest to highest and select emptiest bin
5. Using the solution of BPP for charging, CB, as a guide for amount of energy stored already place energy blocks charged in earlier time steps in this bin until the next block cannot fit without exceeding capacity then move to the next emptiest bin and repeat this process until the energy blocks are used up
6. DCB represents the new demand profile during the discharging phase of the ESS and gives the optimal discharging periods and amounts for peak demand reduction.

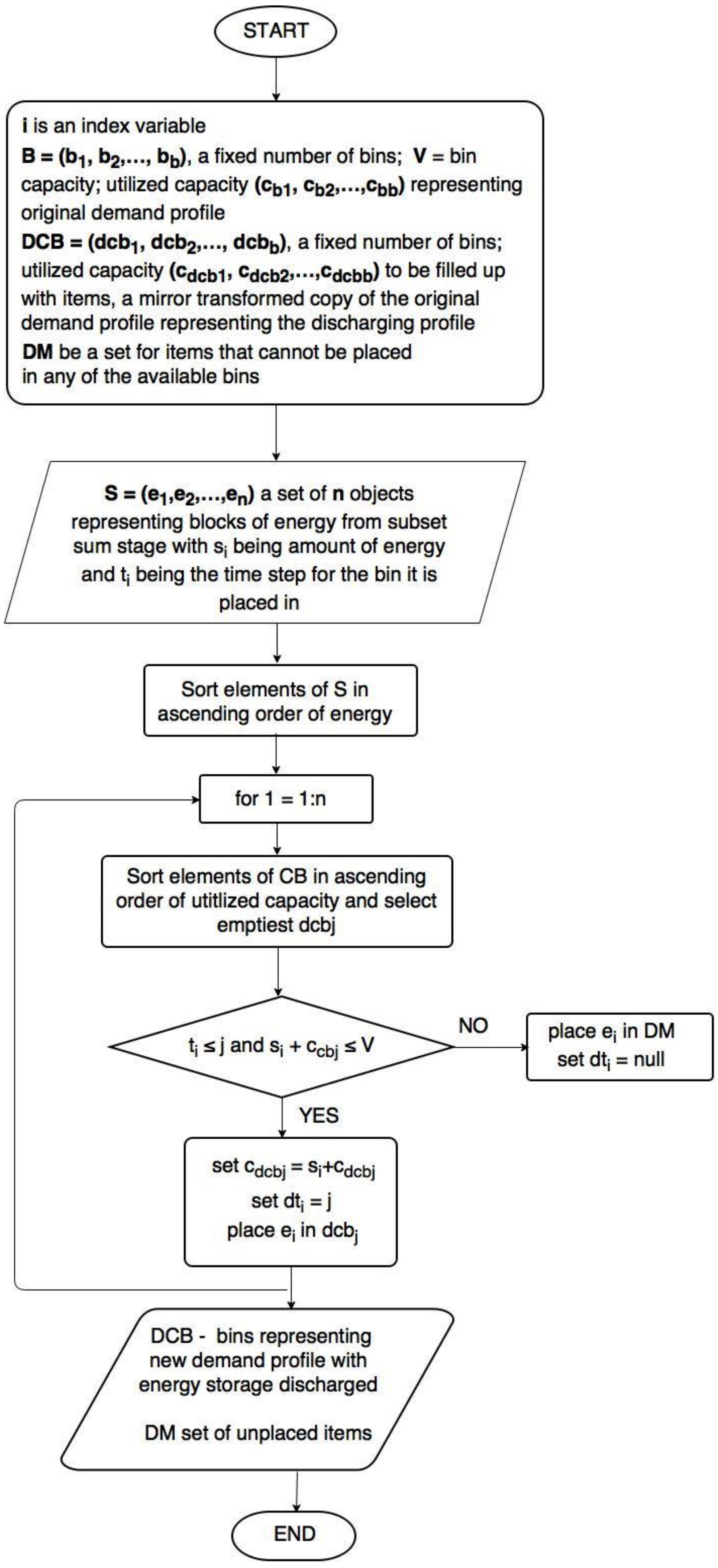

START
i is an index variable
B = (b1, b2,…, bb), a fixed number of bins; V = bin capacity; utilized capacity (cb1, cb2,…,cbb) representing original demand profile
DCB = (dcb1, dcb2,…, dcbb), a fixed number of bins; utilized capacity (cdcb1, cdcb2,…,cdcbb) to be filled up with items, a mirror transformed copy of the original demand profile representing the discharging profile
DM be a set for items that cannot be placed in any of the available bins
S = (e1,e2,…,en) a set of n objects representing blocks of energy from subset sum stage with si being amount of energy and ti being the time step for the bin it is placed in
Sort elements of S in ascending order of energy
for 1 = 1:n
Sort elements of CB in ascending order of utitlized capacity and select emptiest dcbj
ti ≤ j and si + ccbj ≤ V
NO
place ei in DM set dti = null
YES
set cdcbj = si+cdcbj set dti = j place ei in dcbj
DCB - bins representing new demand profile with energy storage discharged
DM set of unplaced items
END

Figure 3.9. Flowchart for discharging ESS using a solution to the bin-packing problem

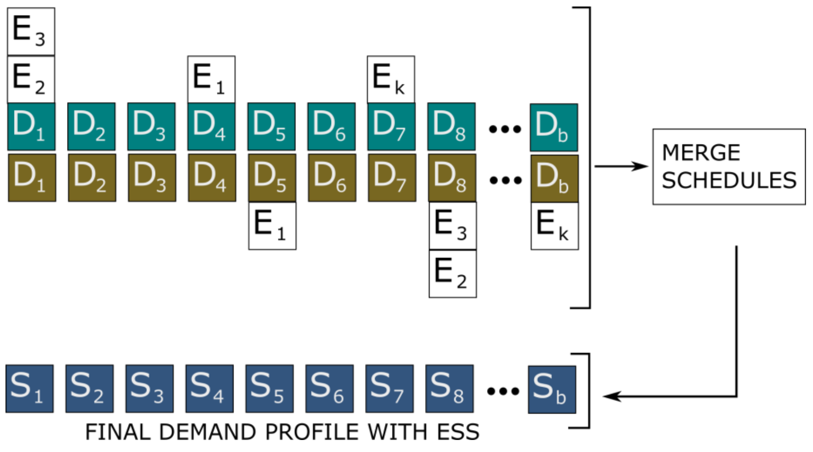
The pseudo-code for the discharging algorithm using the solution to the bin-packing problem and an inverted profile is given in Listing 3.3

Listing 3.3. Pseudo-code for Algorithm for discharging ESS using solution of Demand Profile Bin Packing Problem

```
Let $i$ be an index variable

Let $S = (e_1, e_2, \dots, e_n)$, a set of $n$ objects representing blocks of energy with properties $s_i$ being the amount of energy and $t_i$ the charge time step representing the bin it has already been placed into and $dt_i$ the discharge time step, to be packed as items into a fixed number of bins

Let the size of each item be the amount of energy in the block

Let $B = (b_1, b_2, \dots, b_b)$, a fixed number of bins with a fixed capacity V and utilized capacity $(c_{b1}, c_{b2}, \dots, c_{bn})$ representing the original demand profile

Let $DCB = (dcb_1, dcb_2, \dots, dcb_b)$, a fixed number of bins with a fixed capacity V and utilized capacity $(c_{dcb1}, c_{dcb2}, \dots, c_{dcbb})$ to be filled up with items, a mirror transformed copy of the original demand profile representing the discharging profile

Let DM be a set for items that cannot be placed in any of the available bins

Sort elements of S in ascending order of size

for each $i$ from 1 to $n$:

    Sort elements of DCB with time steps on or after $t_i$ in ascending order of capacity utilized and select emptiest

    Let $dcb_j$ be the selected emptiest bin

    if $t_i$ is not null:
        if ($t_i \le j$ and $s_i + c_{dcbj} \le V$):
            set $c_{dcbj} = s_i + c_{dcbj}$
            set $dt_i = j$
            place $e_i$ in $dcb_j$
        else:
            place $e_i$ in DM
            set $dt_i$ = null
```

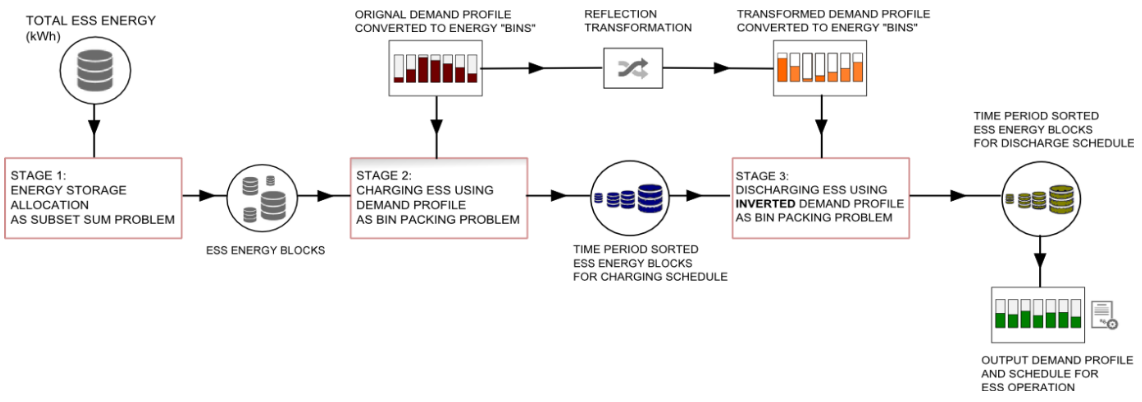
3.4.3 Obtaining solution from merging Charge and Discharge schedules

The charge and discharge schedules are merged by superposition to obtain the final demand profile and schedule of ESS operation (Figure 3.10). This now includes the contribution of the ESS and a schedule for the ESS charge and discharge periods to inform an ANM scheme. The process is illustrated graphically in Figure 3.11 and the flowchart is presented Figure 3.12.



E3
E2
E1
Ek
D1 D2 D3 D4 D5 D6 D7 D8 ••• Db
D1 D2 D3 D4 D5 D6 D7 D8 ••• Db
E1
E3
E2
Ek
MERGE SCHEDULES
S1 S2 S3 S4 S5 S6 S7 S8 ••• Sb
FINAL DEMAND PROFILE WITH ESS

Figure 3.10. Charge and discharge schedules are merged to obtain final demand profile



TOTAL ESS ENERGY (kWh)
STAGE 1: ENERGY STORAGE ALLOCATION AS SUBSET SUM PROBLEM
ESS ENERGY BLOCKS
ORIGNAL DEMAND PROFILE CONVERTED TO ENERGY "BINS"
STAGE 2: CHARGING ESS USING DEMAND PROFILE AS BIN PACKING PROBLEM
REFLECTION TRANSFORMATION
TIME PERIOD SORTED ESS ENERGY BLOCKS FOR CHARGING SCHEDULE
TRANSFORMED DEMAND PROFILE CONVERTED TO ENERGY "BINS"
STAGE 3: DISCHARGING ESS USING INVERTED DEMAND PROFILE AS BIN PACKING PROBLEM
TIME PERIOD SORTED ESS ENERGY BLOCKS FOR DISCHARGE SCHEDULE
OUTPUT DEMAND PROFILE AND SCHEDULE FOR ESS OPERATION

Figure 3.11. Process diagram for combinatorial optimisation algorithm

The steps of the algorithm illustrated in the flowchart and the process diagram are as follows:

1. Using as input the original demand profile which has been binned into a set of bins B and the solution to the subset sum problem for ESS capacity as energy blocks with the bin-packing solutions that give the time period (bin) each block was charged and discharged also noted.
2. Iterate all energy blocks, and for each block iterate through the bins. If energy block was placed in the corresponding bin during the charge period add the capacity of the block to the original utilised capacity. If the energy block was placed in the bin during discharge, subtract it from the original utilised capacity.
3. After iteration set of bins with energy added or subtracted represents demand profile with ESS integration and indicates the schedule for charging and discharging the ESS.

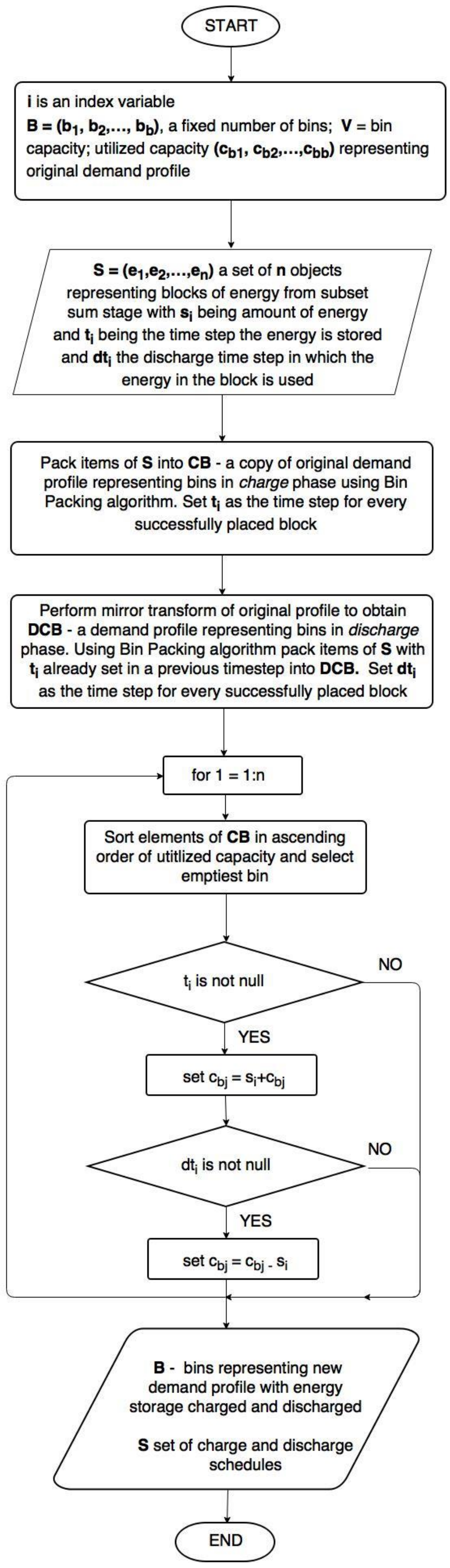

START
i is an index variable
B = (b$_1$, b$_2$,…, b$_b$), a fixed number of bins; **V** = bin capacity; utilized capacity **(c$_{b1}$, c$_{b2}$,…,c$_{bb}$)** representing original demand profile
S = (e$_1$,e$_2$,…,e$_n$) a set of **n** objects representing blocks of energy from subset sum stage with **s$_i$** being amount of energy and **t$_i$** being the time step the energy is stored and **dt$_i$** the discharge time step in which the energy in the block is used
Pack items of **S** into **CB** - a copy of original demand profile representing bins in *charge* phase using Bin Packing algorithm. Set **t$_i$** as the time step for every successfully placed block
Perform mirror transform of original profile to obtain **DCB** - a demand profile representing bins in *discharge* phase. Using Bin Packing algorithm pack items of **S** with **t$_i$** already set in a previous timestep into **DCB.** Set **dt$_i$** as the time step for every successfully placed block
for 1 = 1:n
Sort elements of **CB** in ascending order of utitlized capacity and select emptiest bin
t$_i$ is not null
NO
YES
set c$_{bj}$ = s$_i$+c$_{bj}$
dt$_i$ is not null
NO
YES
set c$_{bj}$ = c$_{bj}$ - s$_i$
B - bins representing new demand profile with energy storage charged and discharged
S set of charge and discharge schedules
END

Figure 3.12. Flowchart for complete process of scheduling ESS using subset sum and bin packing algorithms

The pseudo-code for the merge algorithm to produce the final solution schedule is shown in Listing 3.4.

Listing 3.4. Algorithm pseudo-code for merging charge and discharge schedules to obtain final solution and schedule

```
Let i be an index variable

Let S = ($e_1$,$e_2$,…,$e_n$), a set of n objects representing blocks of energy
with properties $s_i$ being the amount of energy and $t_i$ the charge time
step in which the energy is stored and $dt_i$ the discharge time step in
which the energy in the block is used

Let the size of each item be the amount of energy in the block

Let B = ($b_1$, $b_2$,…, $b_b$), a fixed number of bins with a fixed capacity V
and utilized capacity ($c_{b1}$, $c_{b2}$,…,$c_{bn}$) representing the original demand
profile

for each i from 1 to n:

    if $t_i$ is not null:
        set $c_{bj} = s_i + c_{bj}$

    if $dt_i$ is not null:
        set $c_{bj} = c_{bj} - s_i$

return B, the new demand profile and S, the energy blocks with charge
and discharge schedules
```

3.5 Application of Genetic algorithm optimisation

A Genetic Algorithm (GA) is an evolutionary optimisation algorithm inspired by the theory of natural selection from a population based on fitness for a purpose. It was first used by J.H. Holland in 1975 [57]. From biology a gene may be defined as a “unit of inheritance”, and it is responsible the passage of hereditary traits from parents to offspring. The genes have a fixed position in a *chromosome* in relation to other genes; chromosomes are thread like structures located in the cells of plants and animals that contain instructions on features of the parts of the organism [58]. In humans, for example, genetic diversity is responsible for some easily visible features such as hair colour and eye colour.

Using natural selection organisms adapt to optimise their chances for survival in a given environment, and random mutations occur in the organism's genetic descriptions which are passed to its offspring. If the mutations are helpful the offspring are more likely to survive and reproduce, and if harmful the offspring won't and the traits are eliminated from the population [41].Therefore the individuals that emerge at the end are the fittest and most dominant, hence the term "survival of the fittest".

The principle of GA is to maintain a population of possible solutions to a given problem. These solutions represent chromosomes which are a string of individual characteristics or traits which represent genes. The chromosomes are evaluated and ranked against a fitness function which represents the suitability of the solution to the problem; a biological example could be resistance to a strain of flu – the problem being the flu and the solution being how resistant the organism is.

The fittest chromosomes (parents) in that population are selected for the next generation and are "mated" in a reproductive process known as crossover to produce a new population of possibly fitter offspring. The genes that make up the chromosomes are also subjected to mutation according to a probability known as the mutation rate to possibly improve the fitness of a chromosome and prevent the best solution from being a local optimum. The cycle continues until the termination criteria are met. As the population converges to a local optimum the diversity in the population reduces, therefore it is important to maintain population diversity to attain a global optimum and prevent premature convergence [59]. A proper balancing between diversity and selective pressure is required to reach convergence efficiently.

GA is useful for constrained optimisation problems and therefore has been used in engineering, mathematics and biology applications including image processing, information retrieval, grammar induction, data-mining and natural language processing [59]. It has the advantages of speed, versatility and is relatively easier to implement when compared to classical techniques [8].

GA is occasionally preferred to classical optimisation approaches because it can efficiently handle nonlinear and non-smooth optimisation problems [59] and can usually find a global optimum from a set of solutions. GA differs from classical, derivative-based optimisation in the following ways [60]:

i. The classical methods generate a single point at each iteration and the sequence of points approaches an optimal solution while GAs generate a population of points at each iteration and the best point in the population approaches a solution

ii. Classical algorithms select the next point in the sequence by a deterministic computation while GA selects the next population by computation using random number generators

The algorithm for a simple genetic algorithm applies the following steps:

1. Generate an initial population from a separate function
2. Evaluate the fitness of each member of the population
3. Reproduce my crossover (mating) or mutation to generate new members
4. Using natural selection eliminate the weakest members of the population to incorporate new individual into population thereby producing a new generation
5. Repeat the process for a number of generations or until a suitable fitness criterion is met and then terminate.

GA can be applied to the ESS scheduling method developed in this thesis because it can produce various suitable solutions and it can be encoded in a form suitable for GA easily. The solution demand profile represents a chromosome and the amount of demand in a given interval represents a gene.

To apply it a fitness function is defined and a population of solutions is generated and then optimised via GA.

3.5.1 Fitness function definition and ranking by score

The score for each chromosome is obtained by evaluating the chromosome according to the fitness function defined from the peak shaving and load-levelling D_P and D_L in (3.1) and (3.3). The score is directly proportional to the amount of peak shaving (i.e. difference between new peak and original peak) and inversely proportional to the difference between peak and trough demands. The fitness function in this paper is defined in (3.10).

$$score = \left(D_P(solution) - D_p(original)\right) + \frac{1}{D_L} \tag{3.10}$$

To constrain the solutions to only valid solutions the score is set to zero if the following conditions occur:

i. If the schedule is evaluated to contain more discharge than charge capacity – it is assumed that the ESS is fully discharged at the start of the cycle so it must be charged before discharge

ii. If the total charge or discharge exceeds the capacity of the ESS i.e. constraint in (3.5) must be satisfied

iii. If the ESS demand exceeds the rated power i.e. constraint in (3.6) must be satisfied

This ensures that the schedules with these unrealistic scenarios are eliminated during the GA.

The score function may also be modified to include other objectives for the schedule, for example total energy cost minimisation may be included as an objective.

3.5.2 Populating a generation and evaluating fitness

As a result of the ESS allocation subset sum algorithm different sized ESS units from a constrained random selection will produce various solutions with differing characteristics.

To create the population the allocation and scheduling algorithms are run for the number of members required for the population. When the required population size is reached the chromosomes are evaluated for fitness and ranked in descending order of score to obtain the first generation (Figure 3.13).

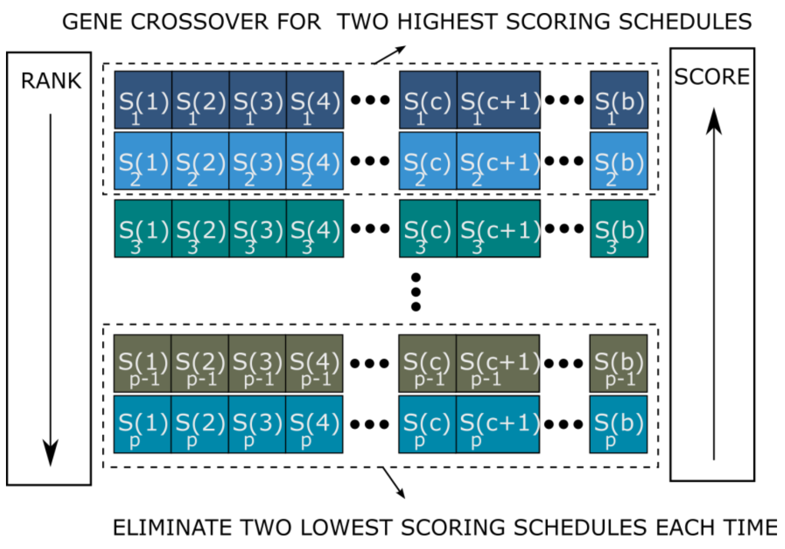

GENE CROSSOVER FOR TWO HIGHEST SCORING SCHEDULES
RANK
SCORE
$S(1)_1$ $S(2)_1$ $S(3)_1$ $S(4)_1$ ••• $S(c)_1$ $S(c+1)_1$ ••• $S(b)_1$
$S(1)_2$ $S(2)_2$ $S(3)_2$ $S(4)_2$ ••• $S(c)_2$ $S(c+1)_2$ ••• $S(b)_2$
$S(1)_3$ $S(2)_3$ $S(3)_3$ $S(4)_3$ ••• $S(c)_3$ $S(c+1)_3$ ••• $S(b)_3$
$S(1)_{p-1}$ $S(2)_{p-1}$ $S(3)_{p-1}$ $S(4)_{p-1}$ ••• $S(c)_{p-1}$ $S(c+1)_{p-1}$ ••• $S(b)_{p-1}$
$S(1)_p$ $S(2)_p$ $S(3)_p$ $S(4)_p$ ••• $S(c)_p$ $S(c+1)_p$ ••• $S(b)_p$
ELIMINATE TWO LOWEST SCORING SCHEDULES EACH TIME

Figure 3.13. Populating, scoring and ranking a generation for crossover and elimination

3.5.3 Crossover

Crossover is implemented by choosing a crossover point along the two fittest chromosomes i.e. the schedules with the highest scores and creating offspring by splitting the schedules at the crossover point and interchanging the resulting splits to create two new schedules as shown in Figure 3.13.

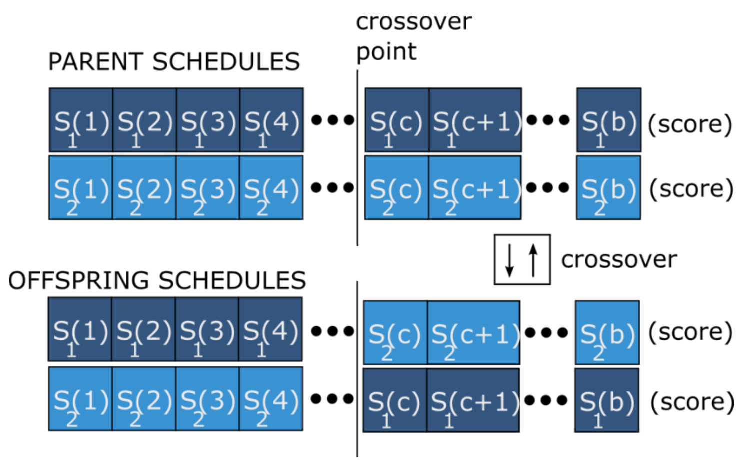


Figure 3.14. Crossover operation to possibly produce fitter schedules from top scoring pair

If the crossover point is *c* and the fittest schedules S_1 and S_2 are concatenations of *b* items according to (3.11) and (3.12).

$$S_1 = S_{11:c} + S_{1c:b} \qquad \textbf{(3.11)}$$

$$S_2 = S_{21:c} + S_{2c:b} \qquad \textbf{(3.12)}$$

Then the offspring O_1 and O_2 after crossover will be (3.13) and (3.14).

$$O_1 = S_{11:c} + S_{2c:b} \qquad \textbf{(3.13)}$$

$$O_2 = S_{21:c} + S_{2c:b} \qquad \textbf{(3.14)}$$

3.5.4 Mutation

The chromosomes in the population are subject to mutation based on probability. The process is independent and changes one of the genes in the chromosome randomly i.e. changes the value of demand in a given period to a randomly selected value.

Each member of the population is evaluated in each generation with a chance of a random demand mutation to unsettle the system and possibly produce an improved schedule (Figure 3.15).

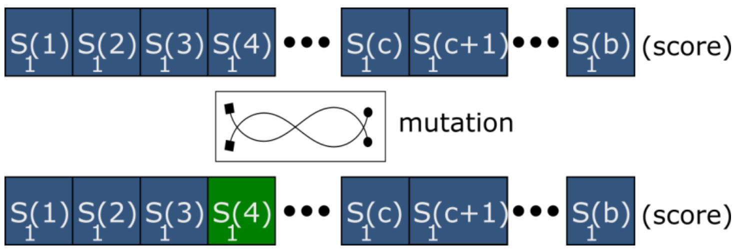


Figure 3.15. Mutation operation to produce a possibly fitter schedule and seek global optima

3.5.5 Selection and Elimination

Selection of chromosomes i.e. the new schedules after crossover and mutation may be done by a different methods such as Proportionate Roulette Wheel, Exponential ranking, Linear Ranking, Tournament Selection etc. [59].

The most popular method of Tournament selection is used in this case due to the efficiency and ease of implementation [59]. After each crossover and mutation the fitness of each offspring or mutated chromosome is evaluated and placed back in the population. If the chromosome is found to be fitter than the lowest ranked in the population, it is selected for further processing in the GA while the least fit is eliminated (Figure 3.13).

3.5.6 Termination

The GA process is iterative and must have conditions for termination. The termination conditions used in this case are:

i. Terminate after a given maximum number of generations

ii. Terminate if the top score does not change after a given number of generations even if condition (i) is not met

The schedule with the highest score upon termination is chosen as the most optimal solution.

3.6 Summary

Using the methodology described a number of viable schedules for operating ESS for peak shaving and load-levelling at distribution network level or consumer level.

The schedules are generated by first splitting the ESS into smaller energy blocks that add up to the total capacity using a subset sum algorithm and then placing the energy blocks in a transformed demand profile using a bin-packing algorithm.

The schedules are ready for dispatch after the subset sum and bin-packing algorithm but may be further optimised using Genetic Algorithm optimisation as described.

4 Web-based Power Systems Analysis: Related work and Literature

4.1 Introduction

To perform power systems analysis using software the basic mathematical models that are used in analysis by hand must be converted to software models using suitable representations. The transition from the mathematical models required for power systems analysis into software models is covered in this section to show the concepts that are fundamental to power systems programming.

An electrical network (or power system) is made up of components which are represented by their properties such as the amount of power they consume, produce or transfer and other parameters such as their voltage levels or the electrical current that passes through them. On paper a diagram may be drawn to indicate the connections between these components to serve as a visual aid for carrying out calculations by hand. In software models they must be represented as a dataset that can be analysed by the computer so as to recreate a software object model in its memory.

In using mathematical concepts, the computer program must also be equipped with tools to perform operations for deriving conditions prevalent on the electrical network being studied in the same way a human must understand a minimum level of mathematics to be able to perform the studies required for analysing electrical networks. While a human can easily recognise the concepts and apply the required methods, the computer program must follow a series of pre-defined general steps in an algorithm to select the specific methods to be applied. The computer however has the advantage of speed and accuracy and it can also handle larger networks and easily keep track of references.

As Power systems analysis (PSA) is a broad topic of discussion this thesis will focus on the most used studies in PSA. The power flow study is the primary subject of PSA [18]. It is a study of the steady-state conditions of a network to obtain unknown parameters from a set of known parameters of the network components. These parameters include the power, voltages, and currents among others. The power flow study establishes the “normal” operation of the network and from this study other studies such as fault studies and stability studies are performed. The short-circuit fault is the most common fault that occurs in an electrical network [18]. It occurs when current flows in a direction that it is not meant to flow in, usually as a result of connections such as cables or transmission lines being bridged by objects such as tree branches and this fault may lead to equipment damage.

The power flow and short-circuit fault studies are covered in this thesis from the fundamental concepts leading up to the development of software models in general and then finally for use in web-based PSA software using PHP.

The previous research work and existing software packages related to Web-based power systems analysis (WBPSA) are reviewed to show the methods used, and the concepts that are transferrable to a PHP-based methodology and those that will not be applicable.

4.2 Power systems representation fundamentals

A power system or electrical network is an interconnected network of elements for the generation, transmission, distribution and consumption of electrical energy. These elements may be divided into four major parts: generation, transmission and sub-transmission, distribution and loads [3].

Generators produce the electrical energy from other forms of energy such as heat or kinetic energy using devices. Transmission and distribution systems include Lines, which are conductors that transfer electrical energy from point in space to another; Buses (or nodes) which are common connection points for Lines and other electrical devices; Transformers, which are devices that transfer power from one voltage level to another voltage level with very high efficiency and make transmission of electrical power over long distances possible. Loads are the devices which consume electrical energy, such as lighting, heating and transportation devices.

Figure 4.1 shows a graphical representation of a power system which consists of three buses, a generator, a motor (e.g. a conveyor belt), a line, two transformers and a load (e.g. a house). The network may be analysed using data provided on each of the elements and the amount of power at each point of the network under normal operation or special conditions may be calculated.

For a system to be safe, reliable and economical many analyses must be performed to design and operate it. For a simple system such as the one in Figure 4.1 it can be analysed by hand using mathematical methods. As the system grows larger it becomes more reliable and easier to perform these studies on the condition of the network using computer programs. The characteristics of such computer programs for power systems analysis are described in the next section.

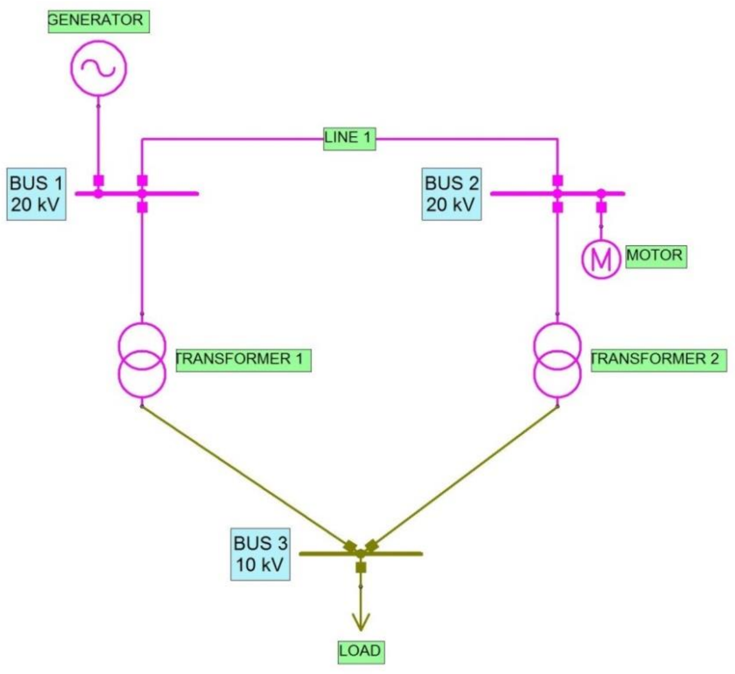


Figure 4.1. A simple 3-bus power system consisting of a generator, transformers and loads

4.3 General Power Systems Analysis and programming: requirements and languages

According to [15] for a programming language to be suitable for PSA, it must be able to perform the following computations:

i. Basic mathematical functions (e.g., exponential, logarithm and trigonometric functions)

ii. Complex numbers

iii. Multi-dimensional arrays (e.g., element by element operations and slicing)

iv. Linear algebra

v. Sparse matrices

vi. Eigenvalue analysis of non-symmetrical matrices

A function is a named block of code that performs a specific task, possibly acting upon a set of values given to it, or parameters, and possibly returning a single value [61]. These functions listed above are required because of the mathematical operations and methods used for modelling power systems.

The list shows the general requirements for most studies and not all of these functions are required for every single study however (i) – (iii) are usually required. The specific requirements for power flow and short circuit studies will be shown in this chapter by reviewing the formulation of the mathematical models for these studies.

Programming languages may be categorised into Legacy system languages (e.g. FORTRAN, C, C++) which can access machine functions in a manner closer to machine language; Modern system languages (e.g. Java, C#, VB, VB.NET) which are more user-friendly and human-readable; General purpose scripting languages (e.g. PHP, Python, JavaScript, Perl); and Scientific-oriented scripting languages (e.g. MatLab and Octave) [15].

The main difference between scripting languages and system programming languages is that the scripting languages are interpreted by a runtime or a web-browser and therefore require these platforms to run on a computer operating system while the system programming languages are compiled and have all the requirements bundled together in the resulting software package.

In terms of structure software based on traditional system programming language is usually a monolithic, stand-alone, self-contained software package. Software developed using a scripting language usually consists of fractal applications or components that work together to provide functionalities [15].

The system programming languages also usually provide more computational functionality than the scripting languages and so they are generally referred to as more “powerful”. The scripting languages may have their core functionality upgraded to enhance their capabilities using libraries. All programming languages have their strengths and weaknesses which are determined by the purpose of the application. In this thesis the focus is on delivering PSA via a web interface and the fitness of these languages for this purpose will be assessed based on how simply they can be deployed on the internet.

The development of a PSA package using either system programming languages or scripting languages requires planning and organisation. The diagram in Figure 4.2 shows the architecture of a general power systems analysis software suite [15] and the functions or modules that work together within the software. The arrows in the figure show the data flows from one component to another or a call to another module to perform an action and optionally return data to the caller. The diagram shows that the software usually includes the following functions:

1. **Data input functions:** These are the programming functions that represent a power system in the memory of a computer program. A *graphics library or Geographical Information System library* may be used to generate the *network diagram* with visual

connections between components which are converted to *input data* or the network data may be supplied directly in an acceptable text format. This input data is fed to a *parser* which interprets the data and creates the model of the network in a format that is usable by the computer program.

2. **Initialization functions:** These functions create the computer models of the network elements by using the data provided for other analysis such as power flow and short circuit studies. For power flow analysis the buses, lines, generators and loads must be defined as a minimum requirement.

3. **Analysis functions:** Figure 4.2 shows *power flow analysis*, *static analyses*, and *dynamic analyses* in the blocks. Different methods exist for any of these analyses which focus on either a steady-state operation of a network or a time-changing analysis of the network.

4. **Output functions:** All PSA packages must have a format for displaying results. It may be in a visual format in the graphical network diagram or in tables and plots. They may also be exported in results files that can be opened in other applications such as spread sheet software.

The implementation of these functions is dependent on the programmer's preference. They may be implemented in a *procedural* style or an *object-oriented* style. Using the procedural style a computer processes instructions sequentially line by line [26]. This is the style of most of the legacy system programming languages such as FORTRAN. As programs become more complex it becomes difficult to manage them using a procedural style; they reach practical functionality limits, changes in logic must be repeated in every part of the program and it becomes difficult to correct mistakes. Object-oriented programming (OOP) addresses these issues by applying a modular approach to programming and grouping functions into concepts known as classes which can be reused.

A class is the fundamental building block of OOP. A variable associated with a class is known as a *property*, and a function associated with a class is known as a *method*. An object is an instance or occurrence of a class [61]. To use a car as an example of a class, there are many types of cars but there is only one physical vehicular concept of mobility referred to in general as a car. A "van object" is an instance of a car, and a "sedan object" is another instance of a car; its weight and colour are its *properties* while actions such as acceleration, braking and steering are its *methods*.

OOP introduces concepts such as *inheritance* – which bestows objects or subclasses of a parent class with its methods and properties; *encapsulation* – which prevents modifications to members of a class from outside the class by making a class self-contained; and *polymorphism* – which allows the same names for functions that play similar roles in different classes, for example a human head and a horse's head have different shapes but essentially play the same role and can both be referred to as heads. In a PSA programming context "Power" in a generator refers to an output while in a load it refers to an input.

Developing a modular framework for PSA is much easier using OOP as a result of the features and benefits it provides. Using OOP for PSA according to the system architecture in Figure 4.2 will involve developing classes that implement the mathematical computations i.e. solvers that carry out the PSA according to fundamental principles and classes for administration of the analysis such as reading input data, displaying results and managing settings for the software application.

The implementation of the PSA software may be done using any language that meets the requirements outlined in this section. These PSA tools may be used on desktop and laptop computers, and more recently on the internet. The focus of this thesis is on the use of these tools on the internet via a web-browser interface.

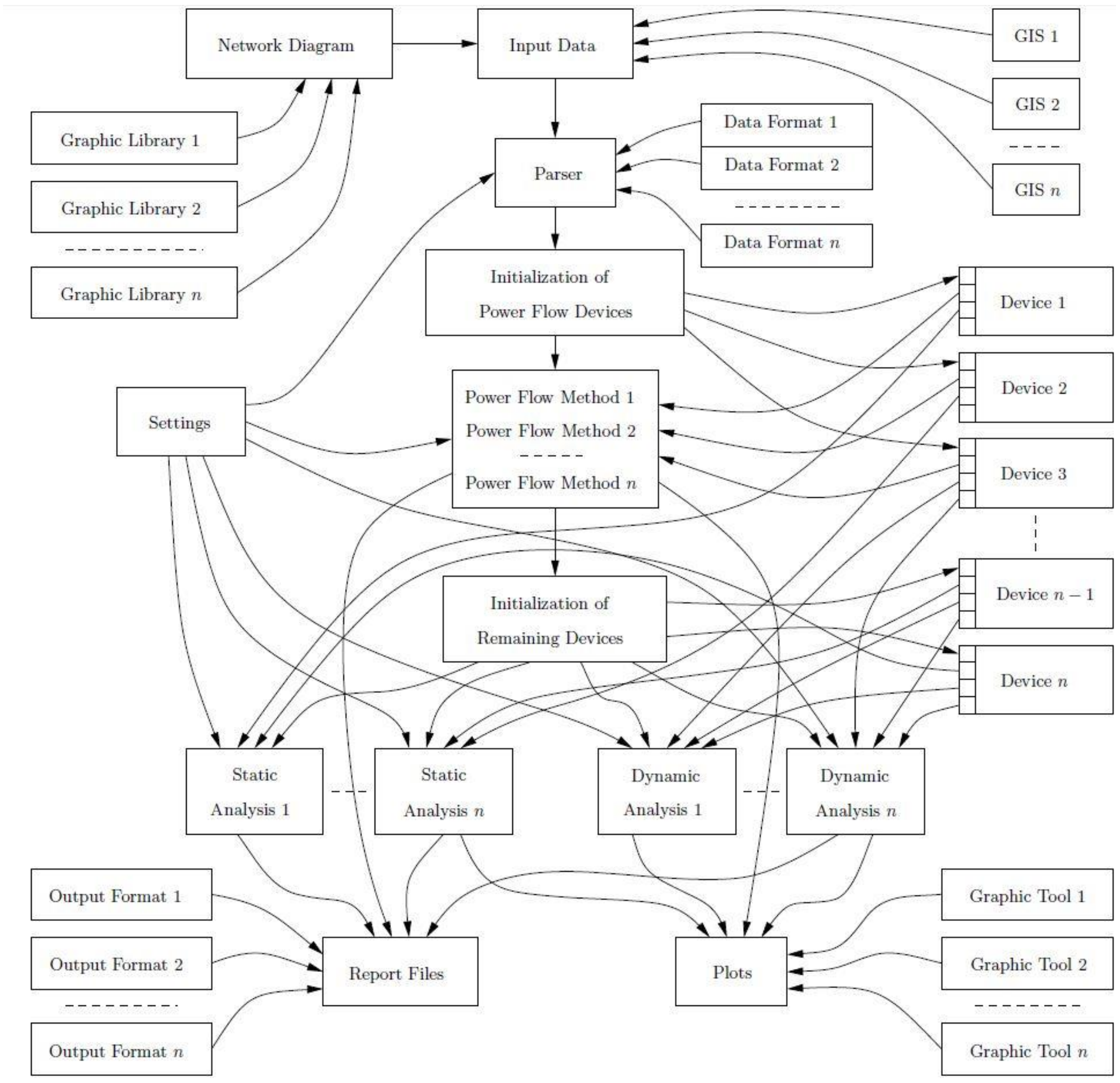


Figure 4.2. System Architecture for a general purpose

4.4 Web-based Power Systems Analysis using a 3-tier framework

The World Wide Web (WWW, or simply Web) is an information space in which the items of interest, referred to as resources are identified by global identifiers called Uniform Resource Identifiers (URI) [62]. The Internet is a global system of interconnected computer networks that interchange data by packet switching using the standardized Internet Protocol Suite. Packets are small bundles of data that are sent over lines of communication. The web allows computer users to locate and view multimedia-based documents over the internet. Previously computer applications and software packages ran on computers that were not connected to one another, but now the internet allows applications to communicate across several computers via the web by mixing computing and communications technologies [21].

Computers connected to the Web are called clients and servers which interact with each other by sending packets of data as requests and responses as shown in Figure 4.3 [63]. Clients are the typical web user's internet connected devices such as personal computers and mobile phones which have web-accessing software such as web-browsers Microsoft Internet Explorer™, Google Chrome™, Mozilla Firefox™. Servers are computers that store web pages, documents, and software that respond to requests from the client with copies of web pages that are downloaded and displayed in the user's web browser. The web server may also respond to requests with data sets or perform an action on the server computer after receiving the request. This is referred to as the server providing a service.

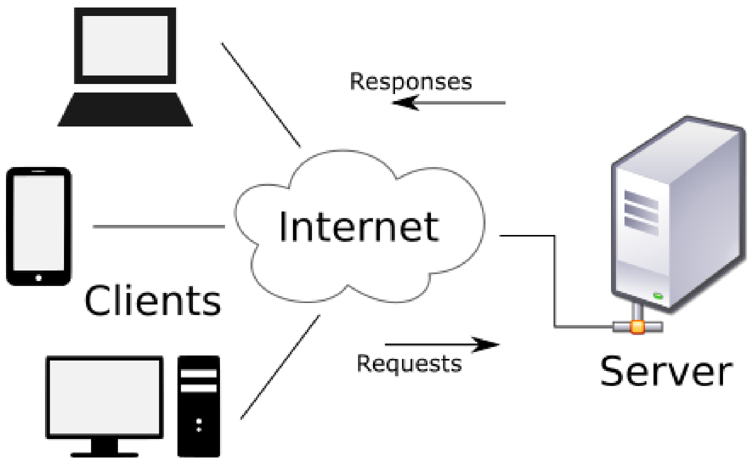


Figure 4.3. World Wide Web (WWW or Web) interaction between Server and Client computers

Web-based simulation (WBS), analysis and remote control of systems via a web browser is increasingly becoming relevant and even necessary in some cases with the rise of cloud computing, Software-as-a-Service (SaaS) and smart grid technologies [64][65]. The information and communications technology infrastructure that enable this service oriented architecture of software have evolved over time, and so have the programming languages that are used to develop these applications.

The context of web-based simulation in this thesis is the definition in [5] as the use of resources and technologies offered by the world-wide-web (WWW) for interaction with client (web browser) and server (remote computer) modelling and simulation tools. Furthermore the definition excludes simulation packages that are downloaded from a server to a local computer and executed independent of the web browser, emphasizing that a browser always has to play an active role in the modelling or simulation process, either as a graphical interface or, additionally, as a container for the simulation numerical engine [5].

Web servers typically run scripts written in ASP, ASP.NET, JSP, Perl, PHP, etc. [28] which are designed primarily for text-based communication using Hypertext Transfer Protocols (HTTP) while simulation packages are built using more powerful languages such as C, C++, C#, Java, etc. [24], [66] because they have a wider range of tools to perform the mathematical analysis required and produce results in a relatively short amount of time. However more recent versions of the web server languages have been upgraded to provide tools for more complex computation such as Object Oriented Programming [26]. The web programming languages are interpreted by the web server software such as Apache or NginX web servers.

For the purpose of carrying out PSA via a web-browser (e.g. Internet Explorer, Mozilla Firefox, Google Chrome etc.) a 3-tier framework is the most used setup as shown in Figure 4.4. One tier consists of the remote simulation server which runs software that simulates a model of the network and performs calculations, the second tier is the web server that handles communication between the other tiers, and the third tier is the web browser for input of parameters and displaying results.

The methodologies for WBPSA in previous research work usually adapt this pattern as will be seen in detail. Active Server Pages (ASP) and ASP.NET are used to process requests and responses in [67]–[69] between the simulation server running C# programs and the web browser. Java Server Pages (JSP) are used in [65] to connect a legacy simulation application based on Fortran to Java applets running in the web browser.

The same architecture is applied in commercial and open source software. NEPLAN 360 [70] is the leader in this category and provides a robust application via the web interface which has all the key features of the desktop version along with an Application Programming Interface for third party access. The legacy application runs on the simulation server and is served through a cloud computing interface to a Microsoft Silverlight™ plugin in the web browser which allows interactivity. There is no indication that the actual simulation is done via a web server script and therefore it also has 3-tier architecture. MATLAB power systems solvers such as MATPOWER [71] may also be accessed on the web by running them on an application server with MATLAB installed and then using an intermediate scripting language such as ASP.NET to communicate with the web browser [72]. Another web-based cloud application is InterPSS [73] which is based on Google Drive ™ spread sheets for data input and results output, and a Google Apps™ script communicates with a Java based simulation engine.

The 3-tier architecture is implemented for practical reasons. One reason is that the simulation software packages which are not ready for internet modelling use a web server as an interface. The only requirement for web access will now be an upgrade of communication and data transfer protocols not the simulation methods. Another practical reason for the 3-tier architecture is that the programming languages used for building websites are not as powerful or as purposeful for computation as those used for building simulation software.

Some of the drawbacks of the 3-tier architecture in general are as follows:

i. There are multiple points at which failure can occur in the process. The multitier nature with different applications in different programming languages handling data and passing it on to the next tier allows for failure at any of these points to cause a breakdown in the system

ii. Some of the legacy system programming languages are premium packages with expensive licensing fees which makes them unsuitable for research, education and smaller organisations

iii. The server infrastructure requirements are higher for running the web server and application server for simulations. A slimmer framework with fewer tiers will reduce the memory and space requirements.

iv. Website developers will need to understand the programming languages used in the simulation engine to some extent if the package does not provide simple access protocols through a programming interface. This presents a steep learning curve for the developers.

The drawbacks of some of the languages used for PSA in the simulation engine tier are documented in [15]. FORTRAN and C are not convenient for complex modular projects; Java promotes a closed software model that should be avoided in education and research. MATLAB is the most used language in education and research, along with languages such as Mathematica, Octave and R. The drawbacks highlighted for these languages include being designed to be very specific to mathematical tasks such as matrix computation and statistics but can be inconvenient for simple data types like tuples, lists and hashes; OOP may be difficult or impossible to implement; they usually are proprietary software and therefore may be expensive to use, with third-party interpreters being slow. Because of these drawbacks [15] makes arguments for why scripting languages such as Python and PHP should be used for research and education programming of power systems.

The methodology in this thesis will propose a 2-tier framework using PHP but before doing so the implementation of the 3-tier frameworks in various open source and commercial software and in research work will be reviewed in the next sections.

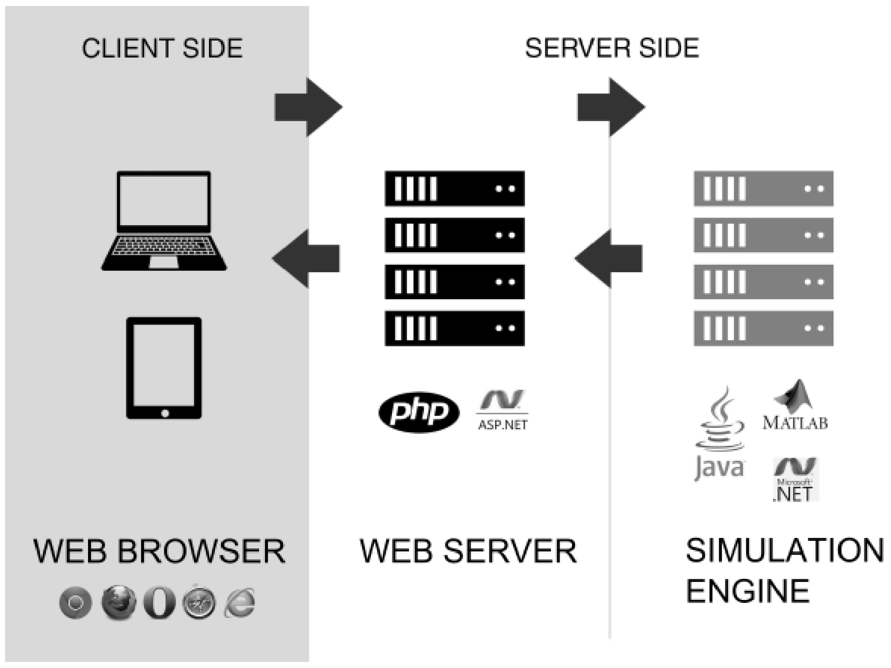


Figure 4.4. 3-tier architecture for Web-based Power Systems Analysis

4.5 Commercial and open source software for web-based PSA

The most common commercial and open source Power Systems Simulations software packages are listed by Open Electrical in [66]. The IEEE Power System Analysis, Computing, and Economics working group also sponsors a Taskforce for Open Source Software for Power Systems [25].

The packages in these detailed lists that explicitly have Web-based interfaces or a module that can allow web access are InterPSS and NEPLAN 360 from BCP Switzerland. SimPowerSystems from MathWorks extends MATLAB’s Simulink and MatPower, which was developed by members of the Power Systems Engineering Research Centre of the Cornell University, is a set of MATLAB files with Power Systems analysis functions [74]. MATLAB programs can now be deployed as web applications [72] , hence it follows that MatPower and SimPowerSystems solutions can also be deployed online and may be classified as a Web-based solution. An overview of the web related characteristics of each of these packages is covered in this section.

4.5.1 NEPLAN 360

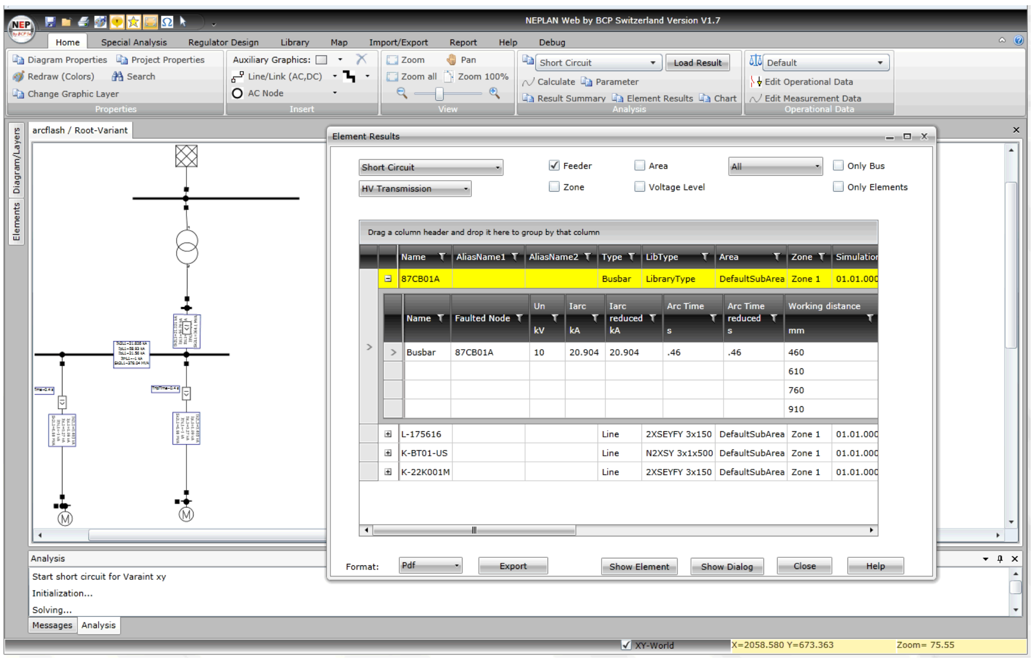
NEPLAN 360 is a power system analysis tool that can be operated from inside a web browser and is licensed by Neplan AG. According to the NEPLAN 360 Website it is the first fully browser based power system analysis tool on the market and therefore offers all advantages of cloud computing [75].

The calculation modules have the same characteristics as the ones in the desktop version of NEPLAN and it handles AC and DC networks in the same manner as the desktop version [75]. It also provides a similar Graphical User Interface (GUI) with drag-and-drop functionality for building electrical networks as shown in Figure 4.5. The multi-user database is based on MS-SQL or Oracle and built to handle large networks and also a large number of users.

The web browser access to the NEPLAN software which runs on a remote server is via a Flash plugin or Microsoft Silverlight plugin that is downloaded to the web browser to communicate with the server and display tools for interaction. Flash and Silverlight extend the functionality available to web browsers so as to handle animations, video playback and improved interactivity which are useful for GUI design of electrical networks. However these browser plugins are being deprecated and are not supported in some modern web browsers as these functions are now available in the browsers as a standard.

NEPLAN 360 is also accessible through Web Services and allows integration with external GIS, SCADA or Smart Grid application. The software therefore can have the function of a software service (SaS).

NEPLAN 360 implements the 3-tier architecture as the PSA computation is done by the legacy application running on their online platform and then sent to the browser using a web server scripting language as a package downloaded through the Flash or Silverlight plugin.



Short Circuit | HV Transmission | ✓ Feeder | Zone | Area | Voltage Level | All | Only Bus | Only Elements

Drag a column header and drop it here to group by that column

Name	AliasName1	AliasName2	Type	LibType	Area	Zone	Simulatio
87CB01A			Busbar	LibraryType	DefaultSubArea	Zone 1	01.01.000

Name	Faulted Node	Un kV	Iarc kA	Iarc reduced kA	Arc Time s	Arc Time reduced s	Working distance mm
Busbar	87CB01A	10	20.904	20.904	.46	.46	460
							610
							760
							910

Name	AliasName1	AliasName2	Type	LibType	Area	Zone	Simulatio
L-175616			Line	2XSEYFY 3x150	DefaultSubArea	Zone 1	01.01.000
K-BT01-US			Line	N2XSY 3x1x500	DefaultSubArea	Zone 1	01.01.000
K-22K001M			Line	2XSEYFY 3x150	DefaultSubArea	Zone 1	01.01.000

```
Start short circuit for Varaint xy
Initialization...
Solving...
```

Figure 4.5. NEPLAN 360 Graphical User Interface

4.5.2 InterPSS

InterPSS stands for Internet Technology Based Power Systems Simulator. It was developed using Java, XML and the Eclipse IDE. The web-access is available in InterPSS 2.0 which is completely cloud based [73].

The High Level Application diagram of the components is as shown in Figure 4.6 [73] and it shows how data input and results output are implemented at the browser level on client devices such as personal computers and mobile devices. The web browser accesses the PSA using a Google Drive spread sheet template and a set of common shared libraries hosted on the InterPSS account. Users can copy to their Google Drive Accounts, open and edit in a Web browser. The simulation engine runs in on a cloud server and receives a simulation request via a Google App Script (based on JavaScript) embedded in the spreadsheet, carries out the processing and sends the result back to the spreadsheet where it can be stored.

It is a 3-tier System with a Java Program as the Simulation Engine, and the Google App Script and Google Drive Spreadsheet as the data transport tier via a Google Web Service, and the web browser for user interaction via the spreadsheet.

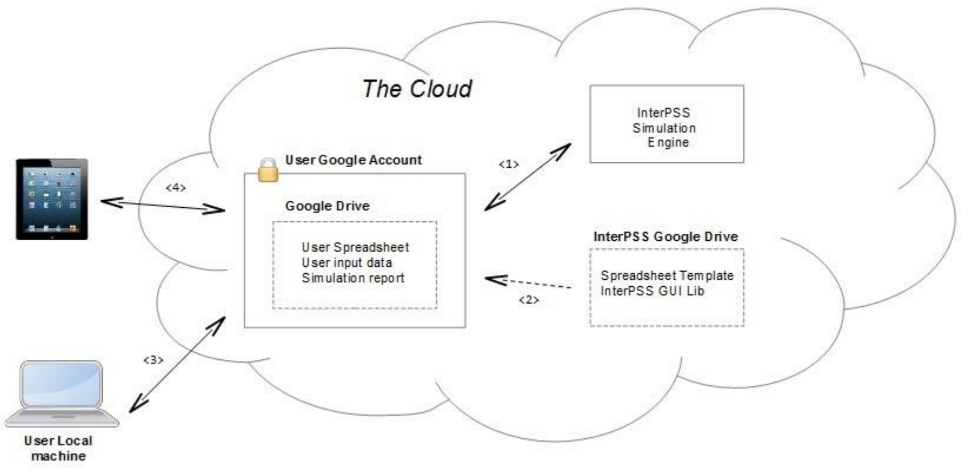


Figure 4.6. InterPSS High Level Architecture

4.5.3 MATLAB based Systems: SimPowerSystems, MATPOWER and others

The Web implementation of SimPowerSystems, MatPower and other PSA tools written in MATLAB is inferred from the capability of MATLAB files to be run from Web Applications. This is possible as MATLAB running on a server can be invoked as an Automation server from any language that supports Component Object Model (COM), so Web applications can use languages such as ASP.NET[76], VBScript and JavaScript [72] to call MATLAB functions.

This is also a 3-tier methodology as shown in Figure 4.7 with the MATLAB application residing on a server being called from an intermediate language such as PHP or ASP.NET on a web server or in the web browser[77].

The web server is in the IT infrastructure block in Figure 4.7 and allows communication of data between the users and the application server via the Web. In the application server tier resides the MATLAB production server that serves as the simulation engine and does the computations. A request broker formats and forwards the requests to the MATLAB application which may also be compiled into different languages such as C++, Java or Python. The program manager coordinates the operations in the application.

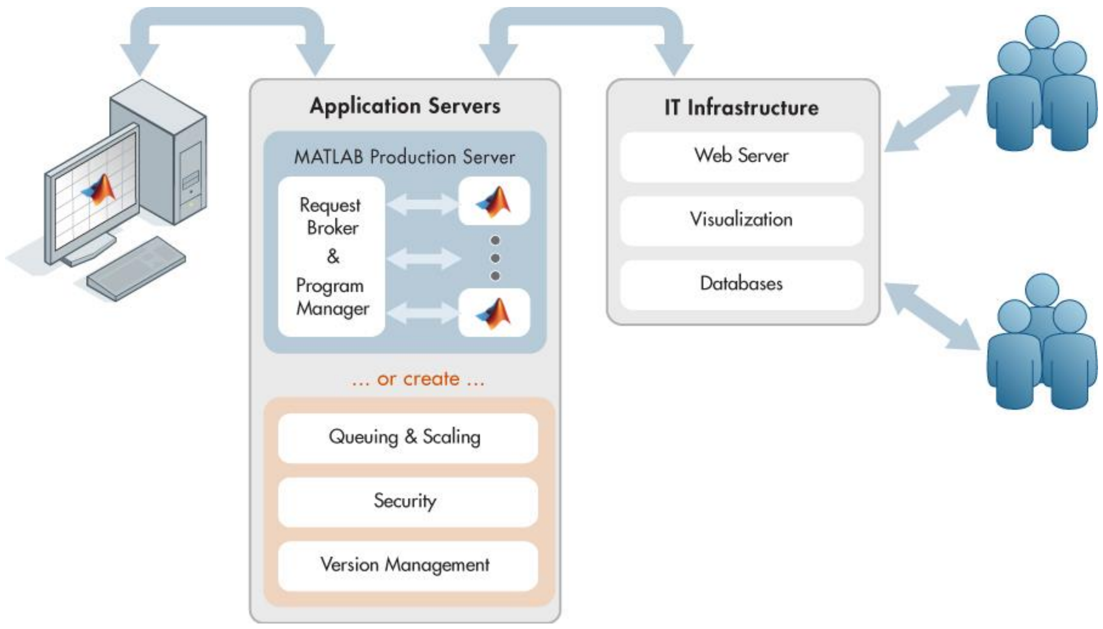


Figure 4.7. MATLAB deployment for web applications using the compiler SDK

4.6 Web-based PSA in previous research work

There have been implementations of Web-based PSS in previous research work, perhaps not as many as one would expect given the ubiquitous nature of Web-based systems recently.

Leou and Gaing in [67] use Active Server Pages (ASP), which is a web programming language similar to PHP and a predecessor of ASP.NET, to call functions in modules programmed in Visual Basic programming; therefore implementing a 3-tier architecture as previously described.

In [65] S. Chen and F.Y. Lu describe a system based on a Model-View-Controller framework (MVC) that uses a Java 2 Platform Enterprise Edition (J2EE) architecture to connect to an existing legacy system by using Java Server Pages (JSP) in the server, applets in the web browser and Fortran based simulation routines as the Simulation engine in a 3+ multi-tier system with the web server acting as a gateway. This is shown in Figure 4.8 where it can be seen that the web server acts as a data and command transport layer and all the PSA computation functions are carried out in the legacy systems tier.

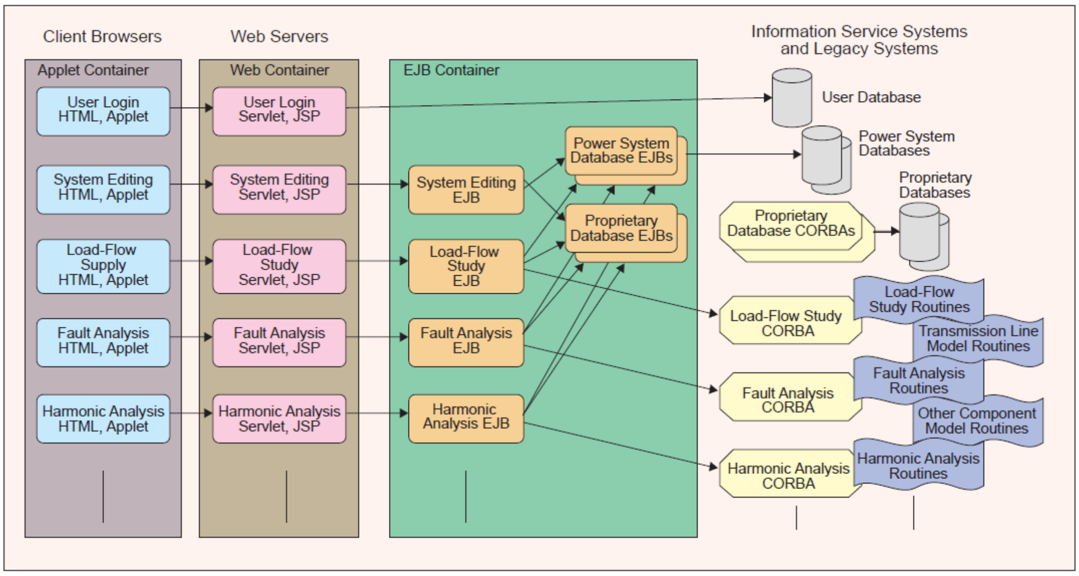

Client Browsers
Web Servers
Information Service Systems and Legacy Systems
Applet Container
Web Container
EJB Container
User Login HTML, Applet
User Login Servlet, JSP
User Database
System Editing HTML, Applet
System Editing Servlet, JSP
System Editing EJB
Power System Database EJBs
Power System Databases
Proprietary Databases
Proprietary Database EJBs
Proprietary Database CORBAs
Load-Flow Supply HTML, Applet
Load-Flow Study Servlet, JSP
Load-Flow Study EJB
Load-Flow Study Routines
Load-Flow Study CORBA
Transmission Line Model Routines
Fault Analysis HTML, Applet
Fault Analysis Servlet, JSP
Fault Analysis EJB
Fault Analysis Routines
Fault Analysis CORBA
Other Component Model Routines
Harmonic Analysis HTML, Applet
Harmonic Analysis Servlet, JSP
Harmonic Analysis EJB
Harmonic Analysis CORBA
Harmonic Analysis Routines

Figure 4.8. Web-based Power systems analysis architecture from Chen and Lu

J. Yang, F. Lin and Y.Fu implement a .NET framework system for micro power system design [68]. The application layer uses the C# based assemblies and Dynamic Link Libraries (DLLs) for the simulations and ASP.NET for serving results to the web browser.

Shaoqiong Tan et al also implement a Web-based simulator using a stack comprising ASP.NET for the web server programming and C# language for the simulation engine in [69]. They also indicate that C# is an evolution of C and C++, which is designed for building a wide range of enterprise applications that run on the .NET Framework.

Hong Chen et al use Java programming for a SCADA system over Local Area Networks and Internet, with a Java Applet for a GUI [78]. This system isn't browser based and outside the definition of Web-based simulation in this paper.

Milano in [71] develops a MATLAB and Octave-based PSA toolbox by leveraging on Simulink tools and GUI and developing custom functions and algorithms for PSA. This will also be deployable on the web using the 3-tier framework adapted by MATLAB solutions described earlier.

The closest methodology to the one proposed in this thesis is developed by Milano in [15], which is a comprehensive PSA tool which performs analysis including Power Flow analysis, Short circuit fault studies and a number of other PSA studies. It was developed using Python scripting language which is similar to PHP as they are both scripting languages, and it has seen a recent rapid growth in use. Python was not originally designed to run on the web but it now has libraries that allow a web access to python applications in a similar manner to PHP. Although the PSA tool developed in [15] was not originally intended for web use it can also be deployable on the web.

4.7 Power systems studies theory and fundamentals

The formulation of the studies that will be implemented in the WBPSA tool based on PHP proposed in this thesis. The two main studies that are carried out in PSA are the Power Flow study and the Short Circuit study. The theory will be reviewed and the mathematical concepts required to perform these studies will be identified.

4.7.1 Power Flow Study

The Power flow study which is commonly referred to as Load flow forms an important part of power systems analysis. It is used to determine the steady-state operation of an electric power system. It calculates the magnitude of voltage on each bus and the voltage angles and also the real and reactive power flowing through each of the lines and elements in the network. Some quantities are specified before the study and the others are determined as a result of the power flow study.

It essentially provides the “normal” operation of the network under the given conditions, and to detect if any problems will arise so as to address them. The problems could be need for additional power generation, need for placing equipment to regulate voltage levels or need to reinforce lines and equipment. It is necessary for planning, economic scheduling, and control of an existing system as well as planning for its future expansion [3].

In solving a power flow problem the system is assumed to be operating under balanced conditions i.e. a line with multiple conductors has equal load shared on each one and the same voltage flowing through each conductor. A single-phase model is used i.e. the calculations are done considering only one of three possible phases since all the phases are balanced.

Four quantities are associated with each bus: the voltage magnitude $|V|$, the phase angle δ i.e. the angular component of a complex number representation of the voltage, the real (or active) power P and the reactive power Q i.e. the imaginary part of the complex number representation of the power. The buses are classified according to the quantities specified at the beginning of the study as follows [3]:

Slack bus/ Swing bus: This is used as a reference for the calculations and it makes up the difference between scheduled loads and generated power. The voltage magnitude and phase angle are specified.

Load buses: At these buses the active and reactive powers are specified, while the voltage magnitude and phase angle are unknown. They are also known as P-Q buses and power is consumed at these buses.

Regulated buses: These are generator buses and power is injected into the network at these buses. The real power and voltage magnitude are specified, while the phase angles of the voltages and reactive power are to be determined.

Table 4.1 summarises the specified and calculated quantities of each type of bus

Table 4.1. Power flow study bus classification by quantities

Bus type	Specified	Calculated
Slack bus (or swing bus)	$\|V\|,\delta$	P,Q
Load bus or P-Q bus	P,Q	$\|V\|, \delta$
Regulated bus or P-V bus	P,V	Q,δ

Power Flow Equation

Consider a typical bus (or node) of a power network as shown in Figure 4.9 with bus voltages denoted as *V* and line admittances (inverse of resistance) denoted as *y*. Kirchoff's Current Law (KCL) states that the sum of the current entering a bus must be equal to the sum of current leaving the bus.

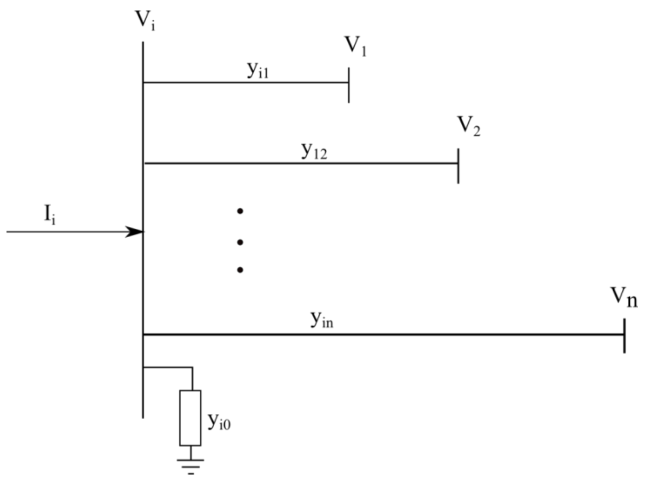


Figure 4.9. Typical bus of a power network

Applying KCL to the bus results in equations (4.1)

$$I_i = V_i \sum_{j=0}^{n} y_{ij} - \sum_{j=1}^{n} y_{ij} V_j \qquad j \neq i \tag{4.1}$$

The real and reactive power at bus *i* is given in (4.2)

$$P_i + jQ_i = V_i I_i^* \tag{4.2}$$

Or

$$I_i = \frac{P_i + jQ_i}{V_i^*} \tag{4.3}$$

Substituting for I_i in (4.1) yields (4.4)

$$\frac{P_i + jQ_i}{V_i^*} = V_i \sum_{j=0}^{n} y_{ij} - \sum_{j=1}^{n} y_{ij} V_j \tag{4.4}$$

The relation in (4.4) is the mathematical formulation of the power flow problem and results in a system of algebraic non-linear equations which must be solved by iterative techniques [3]. There are several techniques that may be used to obtain the power flow solution for the equations based on numerical analysis methods including the Gauss-Seidel (GS) method, the Newton-Raphson (NR) method and the Fast-Decoupled power flow method.

The Newton-Raphson is chosen in this thesis because of its quadratic convergence and it is therefore mathematically superior to the Gauss-Seidel method and is less prone to divergence with ill-conditioned problems (i.e. systems that are sensitive to very small changes resulting in large consequences). The NR method is found to be more efficient and practical for large power systems because the number of iterations required is independent of the system size.

Bus Admittance Matrix

The bus admittance (or Ybus) matrix is one of the fundamental building blocks of power system simulations as it describes the topology and admittances / impedances of an electrical network.

Admittance is the inverse of impedance, which is the measure of the opposition that a circuit presents to a current when a voltage is applied. For example if a voltage is applied to one end of a transmission line the impedance of the line is a measure of the opposition the line presents to the flow of the current through it. It is the complex ratio of voltage to current in an AC circuit.

Take a 2-bus network as shown in Figure 4.10. 2-bus network single line diagram. Applying KCL at each bus the current equations are given in (4.5) – (4.8)

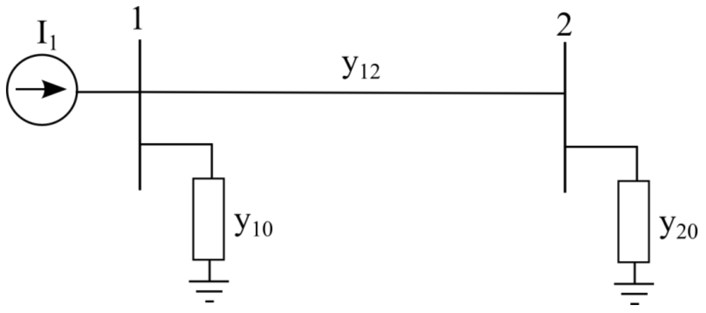


Figure 4.10. 2-bus network single line diagram

$$I_1 = V_1 y_{10} + (V_1 - V_2) y_{12} \tag{4.5}$$

Or

$$I_1 = V_1 (y_{10} + y_{12}) - V_2 y_{12} \tag{4.6}$$

$$0 = V_2 y_{20} + (V_2 - V_1) y_{12} \tag{4.7}$$

Or

$$0 = V_2 (y_{12} + y_{20}) - V_1 y_{12} \tag{4.8}$$

where

$$y_{ij} = \frac{1}{z_{ij}} = \frac{1}{r_{ij}+jx_{ij}} \tag{4.9}$$

Z = impedance; r = resistance; x = reactance

The Ybus matrix is constructed by introducing the self-admittances and the mutual admittances in the network as follows:

Self-admittances (diagonal elements) at bus *k* are the sum of all branch admittances connected to the bus (including shunt admittances to earth) in (4.10)

$$Y_{kk} = \sum_{i=1}^{n} y_{ki} \tag{4.10}$$

Mutual admittances (off-diagonal elements) between buses *k* and *i* are the negative sum of branch admittances connected between buses *k* and *i*:

$$Y_{ik} = -\sum_{i=1 \neq k}^{n} y_{ki} \tag{4.11}$$

From (4.10) and (4.11) the admittances for the 2-bus network are given in (4.12) – (4.14)

$$Y_{11} = y_{10} + y_{12} \tag{4.12}$$

$$Y_{12} = Y_{21} = -y_{12} \tag{4.13}$$

$$Y_{22} = y_{20} + y_{12} \tag{4.14}$$

Substituting the admittances into the bus current equations yields (4.15) – (4.16)

$$I_1 = V_1Y_{11} + V_2Y_{12} \tag{4.15}$$

$$0 = V_2Y_{22} + V_1Y_{21} \tag{4.16}$$

Applying the same principle to an *n* bus system in Matrix form:

$$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1n} \\ Y_{21} & Y_{22} & \dots & Y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \dots & Y_{nn} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} \tag{4.17}$$

$$I_{bus} = Y_{bus}V_{bus} \tag{4.18}$$

where I_{bus} is the vector of the injected bus currents from external sourcess

The current is positive when flowing towards the bus and negative if flowing away from the bus. V_{bus} is the vector of bus voltages measured from the reference node. Y_{bus} is the bus admittance matrix. The diagonal elements of the Y_{bus} matrix are referred to as self-admittances and the off-diagonal elements are mutual admittances. The inverse of the bus admittance matrix is known as the bus impedance matrix, or Z_{bus}.

Newton-Raphson Power Flow

The Newton-Raphson (NR) method is based on the numerical analysis solution of simultaneous non-linear algebraic equations, which is a successive approximation procedure based on an initial estimate of the unknown and the use of Taylor's series expansion [3]. It is applied in PSA as described in this section with the formulation obtained from [3].

For the typical system shown in Figure 4.9 the current entering the ith bus (4.1) can be re-written in terms of the admittance matrix in (4.18) as (4.19)

$$I_i = \sum_{j=1}^{n} Y_{ij} V_j \tag{4.19}$$

Expressing in polar form yields (4.20)

$$I_i = \sum_{j=1}^{n} |Y_{ij}||V_j| \angle \theta_{ij} + \delta_j \tag{4.20}$$

The complex power at bus i is:

$$P_i - jQ_i = V_i^* I_i \tag{4.21}$$

Substituting for Ii from (4.20)

$$P_i - jQ_i = |V_i| \angle - \delta_i \sum_{j=1}^{n} |Y_{ij}||V_j| \angle \theta_{ij} + \delta_j \tag{4.22}$$

Separating real and imaginary parts:

$$P_i = |V_i| \angle - \delta_i \sum_{j=1}^{n} |Y_{ij}||V_j| \angle \theta_{ij} + \delta_j \tag{4.23}$$

$$Q_i = -\sum_{j=1}^{n} |V_i||V_j||Y_{ij}| \sin(\theta_{ij} - \delta_i + \delta_j) \tag{4.24}$$

(4.23) and (4.24) are a set of non-linear algebraic equations in terms of independent variables, voltage magnitude in per unit and phase angle in radians. Each load bus will have both equations (4.23) and (4.24) and each regulated bus will have only (4.24). Expanding them in a Taylor's series expansion for the kth iteration and neglecting higher order terms yields (4.25):

$$\begin{bmatrix} \Delta P_2^{(k)} \\ \vdots \\ \Delta P_n^{(k)} \\ \hline \Delta Q_2^{(k)} \\ \vdots \\ \Delta Q_n^{(k)} \end{bmatrix} = \left[\begin{array}{c|c} \begin{pmatrix} \frac{\partial P_2^{(k)}}{\partial \delta_2^{(k)}} & \cdots & \frac{\partial P_2^{(k)}}{\partial \delta_n^{(k)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial P_n^{(k)}}{\partial \delta_2^{(k)}} & \cdots & \frac{\partial P_n^{(k)}}{\partial \delta_n^{(k)}} \end{pmatrix} & \begin{pmatrix} \frac{\partial P_2^{(k)}}{\partial |V_2|} & \cdots & \frac{\partial P_2^{(k)}}{\partial |V_n|} \\ \vdots & \ddots & \vdots \\ \frac{\partial P_n^{(k)}}{\partial |V_2|} & \cdots & \frac{\partial P_n^{(k)}}{\partial |V_n|} \end{pmatrix} \\ \hline \begin{pmatrix} \frac{\partial Q_2^{(k)}}{\partial \delta_2^{(k)}} & \cdots & \frac{\partial Q_n^{(k)}}{\partial \delta_2^{(k)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial Q_n^{(k)}}{\partial \delta_2^{(k)}} & \cdots & \frac{\partial Q_n^{(k)}}{\partial \delta_n^{(k)}} \end{pmatrix} & \begin{pmatrix} \frac{\partial Q_2^{(k)}}{\partial |V_2|} & \cdots & \frac{\partial Q_2^{(k)}}{\partial |V_n|} \\ \vdots & \ddots & \vdots \\ \frac{\partial Q_n^{(k)}}{\partial |V_2|} & \cdots & \frac{\partial Q_n^{(k)}}{\partial |V_n|} \end{pmatrix} \end{array}\right] \begin{bmatrix} \Delta \delta_2^{(k)} \\ \vdots \\ \Delta \delta_n^{(k)} \\ \hline \Delta \left|V_2^{(k)}\right| \\ \vdots \\ \Delta \left|V_n^{(k)}\right| \end{bmatrix} \quad \textbf{(4.25)}$$

The Jacobian matrix gives the relationship between small changes in voltage angle $\Delta\delta_i^{(k)}$ and voltage magnitude $\Delta|V_i^{(k)}|$ with the small changes in real and reactive power $\Delta P_i^{(k)}$ and $\Delta Q_i^{(k)}$. Elements of the Jacobian matrix are the partial derivatives of (4.23) and (4.24) with respect to $\Delta\delta_i^{(k)}$ and $\Delta|V_i^{(k)}|$. It is expressed in short form as (4.26)

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} \begin{bmatrix} \Delta\delta \\ \Delta|V| \end{bmatrix} \quad \textbf{(4.26)}$$

The diagonal and off-diagonal elements of J_1 are given by (4.27) and (4.28)

$$\frac{\partial P_i}{\partial \delta_i} = \sum_{j \neq i} |V_i||V_j||Y_{ij}| \sin(\theta_{ij} - \delta_i + \delta_j) \quad \textbf{(4.27)}$$

$$\frac{\partial P_i}{\partial \delta_j} = -|V_i||V_j||Y_{ij}| \sin\left(\theta_{ij} - \delta_i + \delta_j\right) \quad j \neq i \quad \textbf{(4.28)}$$

The same approach is used to obtain the diagonal and off-diagonal elements of J_2,J_3 and J_4. The power residuals are the difference between the scheduled and calculated values of power and are given by $\Delta P_i^{(k)}$ and $\Delta Q_i^{(k)}$ in (4.29) and (4.30).

$$\Delta P_i^{(k)} = P_i^{(sch)} - P_i^{(k)} \quad \textbf{(4.29)}$$

$$\Delta Q_i^{(k)} = Q_i^{(sch)} - Q_i^{(k)} \quad \textbf{(4.30)}$$

The new estimates for bus voltages are (4.32) and (4.33)

$$\delta_i^{(k+1)} = \delta_i^{(k)} + \Delta\delta_i^{(k)} \quad \textbf{(4.31)}$$

$$V_i^{(k+1)} = V_i^{(k)} + \Delta|V_i^{(k)}| \quad \textbf{(4.32)}$$

The algorithm for the power flow study using the Newton-Raphson method is outlined as follows:

1. For the Load buses (where P and Q specified) use a flat voltage start i.e. $\left|V_i^{(0)}\right| = 1.0$ and $\left|\delta_i^{(0)}\right| = 0.0$ and for voltage controlled buses (P,V specified) $\left|\delta_i^{(0)}\right| = 0.0$.
2. For Load buses $P_i^{(k)}$ and $Q_i^{(k)}$ are calculated from (4.23) and (4.24) and $\Delta P_i^{(k)}$ and $\Delta Q_i^{(k)}$ are calculated from (4.29) and (4.30).
3. For voltage controlled buses $P_i^{(k)}$ and $\Delta P_i^{(k)}$are calculated from (4.23) and (4.29) respectively.
4. The elements of the Jacobian matrix are calculated.
5. The linear simultaneous equation (4.26) is solved directly by optimally ordered triangle factorization and Gaussian elimination.
6. The new voltage magnitudes and phase angles are computed from (4.31) and (4.32)
7. The process is continued until the residuals $\Delta P_i^{(k)}$ and $\Delta Q_i^{(k)}$ are less than the specified accuracy $\left|\Delta P_i^{(k)}\right| \le \epsilon$ and $\left|\Delta Q_i^{(k)}\right| \le \epsilon$

Line Flows and losses

When the iterative solution for the voltages in each bus is obtained the next step is the computation of line flows and line losses. Figure 4.11 shows a simple 2-bus model with current injected at both buses *i* and *j* which are connected by a line.

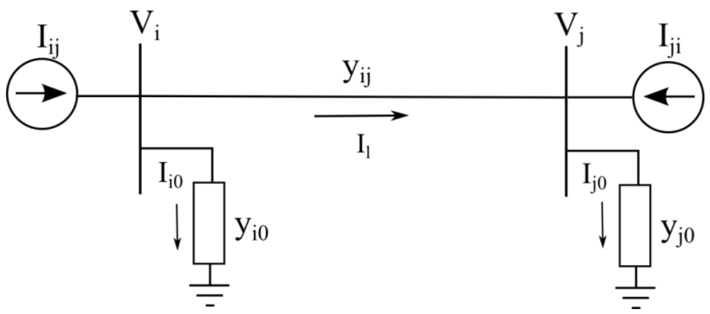


Figure 4.11. 2-bus network with current injection and line flows

The line current I_{ij} at bus I and defined as positive in the direction $i \rightarrow j$ is given by (4.33) and denotes the current flowing from i to j.

$$I_{ij} = I_l + I_{i0} = y_{ij}\left(V_i - V_j\right) + y_{i0}V_i \tag{4.33}$$

And for the other direction $j \rightarrow i$ the line current I_{ji} is given by (4.34)

$$I_{ji} = -I_l + I_{j0} = y_{ij}\left(V_j - V_i\right) + y_{j0}V_j \tag{4.34}$$

The complex power flows S_{ij} from bus i to bus j and Sji from bus j to i are given in (4.35) and (4.36)

$$S_{ij} = V_i I_{ij}^* \tag{4.35}$$

$$S_{ji} = V_j I_{ji}^* \tag{4.36}$$

The power loss is the algebraic sum of (4.35) and (4.36), given in (4.37)

$$S_L = S_{ij} + S_{ji} \tag{4.37}$$

4.7.2 Fault study: Short Circuit Capacity and Systematic Fault Analysis

Fault studies are an important part of power systems analysis consisting of determining the conditions of a network including bus voltages and line currents during various types of faults. A short circuit may be described as an event which occurs when current flows through an unintended path [3]. A common example is when a tree branch falls on an overhead line, bridging the conductors and forming a path for electric current to flow in. They are usually associated with arc flashes, commonly known as “sparks”.

Faults are divided into three-phase balanced faults and unbalanced faults, and the unbalanced faults are divided into single line-to-ground fault, line-to-line fault and double line-to-ground fault. The information gained from the fault study is used to obtain the ratings and settings for protection equipment.

The balanced three-phase fault is covered in this thesis to compute the bus voltages and currents during the fault. The magnitude of the fault currents depend on the internal impedance of the generators plus the impedance of the intervening circuit.

Balanced Three-phase fault

This type of fault occurs across all three phases and is the most severe type of fault to be encountered, although it is infrequent. A fault represents a structural network change equivalent with that of an addition of impedance in the place where the fault occurs.

The short circuit capacity (SCC) is a common measure of the strength of a bus and is defined as the product of the rated bus voltage and the fault current. It is used for determining the physical dimension of a bus bar.

If a fault occurs at bus *k* the short circuit capacity is given by (4.38)

$$SCC = \sqrt{3}\, V_{Lk} I_k(F) \times 10^3 \text{ MVA} \tag{4.38}$$

where the line-to-line voltage is V_{Lk} is in kilovolts and the fault current $I_k(F)$ is in amperes.

The per-unit symmetrical three-phase fault current is given by (4.39)

$$I_k(F) = \frac{V_k(0)}{X_{kk}} \text{ PU} \tag{4.39}$$

where $V_k(0)$ is the per unit prefault bus voltage and X_{kk} is the per unit reactance at the point of the fault. System resistance is neglected and only inductive reactance is used to give a maximum fault current.

The base current is given by

$$I_B = \frac{S_B \times 10^3}{\sqrt{3} V_B} \tag{4.40}$$

Where S_B is the base MVA and $V_B$is the line-to-line base voltage in kilovolts, thus the fault current in amperes is

$$\begin{aligned} I_k(F) &= I_k(F)_{PU}\, I_B \\ &= \frac{V_k(0)}{X_{kk}} \frac{S_B \times 10^3}{\sqrt{3} V_B} \end{aligned} \tag{4.41}$$

Substituting for $I_k(F)$ from (4.41) into (4.38) yields (4.42)

$$SCC = \frac{V_k(0)S_B}{X_{kk}} \frac{V_L}{V_B} \quad \textbf{(4.42)}$$

If the base voltage is equal to the rated voltage i.e. $V_L = V_B$then

$$SCC = \frac{V_k(0)S_B}{X_{kk}} \quad \textbf{(4.43)}$$

Assuming the pre fault bus voltage is 1.0 per unit then the short circuit capacity or short circuit MVA is (4.44)

$$SCC = \frac{S_B}{X_{kk}} \quad \textbf{(4.44)}$$

Systematic Fault Analysis Using Bus Impedance Matrix

The bus impedance matrix or Zbus is the inverse of the admittance matrix and using it the fault current and bus voltages during a fault may be calculated. Consider a typical network with n buses shown in Figure 4.12. A balanced three-phase fault is applied at bus *k* through a fault impedance Z_f.

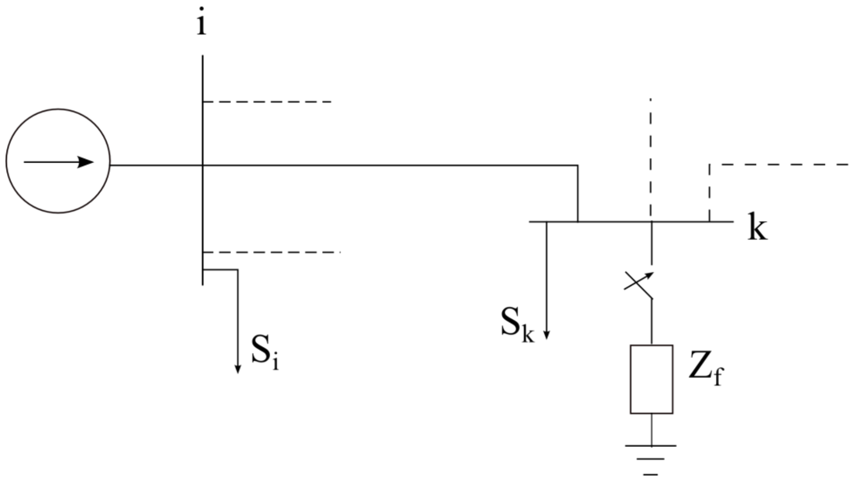


Figure 4.12. Typical bus of a power system

The pre fault voltages are represented by the column vector in (4.45)

$$V_{bus}(0) = \begin{bmatrix} V_1(0) \\ \vdots \\ V_k(0) \\ \vdots \\ V_n(0) \end{bmatrix} \tag{4.45}$$

As the short circuit currents are so much larger than steady state values the latter may be neglected. The bus load may be represented by constant impedance evaluated at the pre fault bus voltage (4.46)

$$Z_{iL} = \frac{|V_i(0)|^2}{S_L^*} \tag{4.46}$$

The changes in the network voltage caused by the fault with impedance Zf is equivalent to those caused by the added voltage $V_k(0)$ with all other sources short circuited. By zeroing all voltage sources and representing all components and loads by their appropriate impedances the Thevenin's circuit (equivalent impendance circuit) is obtained with equivalent impedances at each bus.

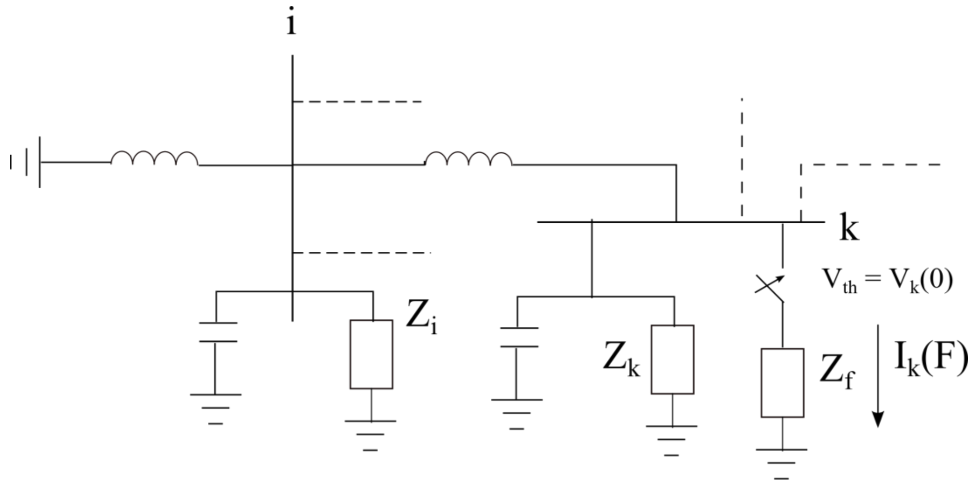


Figure 4.13. Typical bus of a power system with Thevenin equivalent impedances

The bus voltage changes caused by the fault circuit are represented by the column vector in (4.47)

$$\Delta V_{bus} = \begin{bmatrix} \Delta V_1 \\ \vdots \\ \Delta V_k \\ \vdots \\ \Delta V_n \end{bmatrix} \tag{4.47}$$

Bus voltages during the fault are obtained by superposition of the pre-voltages and the changes in bus voltages

$$V_{bus}(F) = V_{bus}(0) + \Delta V_{bus} \tag{4.48}$$

Using the node voltage equations derived in the power flow study for bus currents (4.18) and the mutual admittances (4.11), from the Thevenin's circuit the current entering every bus is zero except the faulted bus thus the nodal equation applied to Figure 4.13 is (4.49)

$$\begin{bmatrix} 0 \\ \vdots \\ -I_k(F) \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} Y_{11} & \dots & Y_{12} & \dots & Y_{1n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_{k1} & \dots & Y_{kk} & \dots & Y_{kn} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_{n1} & \dots & Y_{n2} & \dots & Y_{nn} \end{bmatrix} \begin{bmatrix} \Delta V_1 \\ \vdots \\ \Delta V_2 \\ \vdots \\ \Delta V_n \end{bmatrix} \tag{4.49}$$

Or

$$I_{bus}(F) = Y_{bus} \Delta V_{bus} \tag{4.50}$$

Solving for ΔV_{bus}

$$\Delta V_{bus} = Z_{bus} I_{bus}(F) \tag{4.51}$$

Where

$$Z_{bus} = Y_{bus}^{-1} \tag{4.52}$$

Bus voltage during the fault (4.47) becomes

$$V_{bus}(F) = V_{bus}(0) + Z_{bus} I_{bus}(F) \tag{4.53}$$

Since there is only one non-zero element in the matrix, the *k*th equation becomes:

$$V_k(F) = V_k(0) - Z_{kk} I_k(F) \tag{4.54}$$

From the Thevenin circuit in Figure 4.13:

$$V_k(F) = Z_f I_k(F) \tag{4.55}$$

substituting for $V_k(F)$ in (4.54)

$$I_k(F) = \frac{V_k(0)}{Z_{kk} + Z_f} \tag{4.56}$$

Thus at a faulted bus k only the Z_{kk} element of the bus impedance matrix is needed. Writing the equation for the *i*th bus according to the matrix equations and (4.54):

$$V_i(F) = V_i(0) - Z_{ik} I_k(F) \tag{4.57}$$

And bus voltage during fault after substituting for $I_k(F)$ will be

$$V_i(F) = V_i(0) - V_k(0) \frac{Z_{ik}}{Z_{kk} + Z_f} \tag{4.58}$$

With the knowledge of bus voltages the bus currents in all the lines can be calculated:

$$I_{ij}(F) = \frac{V_i(F) - V_j(F)}{z_{ij}} \tag{4.59}$$

Hence the steps for the three-phase balanced fault calculation are as follows:

1. Run power flow on network and obtain pre-fault bus voltages
2. Compute bus impedance matrix and obtain self-impedance at faulted node
3. Compute fault current using pre fault voltage, bus self-impedance and fault impedance (4.56)
4. Compute fault voltages using fault current derived (4.57) and (4.58)
5. Compute

4.8 Mathematical Modelling Requirements in PSA computing language

From the preceding formulation of the fundamental principles for the Power flow study and the Short-Circuit study, to perform PSA in a computer programming language the following mathematical operations will have to be native to the language or provided by a library.

1. Matrix Operations – for admittance and impedance matrix formation, manipulation and solutions

2. Complex Number / Polar form conversions – for real and reactive power and complex notation of voltages and other quantities

3. Vector and Tuple operations – for solutions of matrix operations

4. Trigonometric functions – sine, cosine etc.

5. Iteration functions – for non-linear programming iterations

5 PHP application library for web-based power systems analysis

5.1 Introduction

This section describes the methodology implemented in building an application library for Web-based Power Systems Simulation using PHP programming and provides a documentation of its functions. The simulation engine performs a load flow analysis on a given network using the Newton-Raphson method according to the procedure outlined in [3] to obtain unknown bus voltage magnitudes and angles, and also the real and reactive power magnitudes for a given network. It also computes the line flows by first computing line currents and then the power flowing in each line as well as its direction. The library also performs short-circuit fault studies to determine the short circuit currents flowing in a network.

A review of the advantages and disadvantages of using PHP in general and for power systems analysis is presented before the system architecture to show the different components of the application and their interactions are outlined in this section. It will show some concepts which are not native to PHP but required for power systems analysis are implemented and then break down the library into its modules and how they interact.

The methodology will be presented with more emphasis on application design and architecture, flowcharts, and interaction of components rather than the PHP code used to implement the concepts. Lines of code and programming will only be presented in snippets when relevant due to the extensive amount of code used for the entire library.

5.2 Advantages and Disadvantages of using a PHP simulation engine for Power Systems Simulation

The merits and demerits of PHP are discussed widely on the internet in detail, for example [79] reviews some of the design flaws of PHP as a programming language as compared to other languages while [22] highlights its benefits in terms of flexibility, availability of functions, etc. Some commentators also point out that some reputed organizations such as Facebook™, Wikipedia™ and Harvard University use PHP [80] . As with any other programming language, the purpose determines what counts as an advantage or disadvantage. In relation to a web based simulation engine, some of the advantages and disadvantages of using PHP for power systems analysis are as follows:

5.2.1 Advantages of using PHP for a power systems simulation library

Platform flexibility

Because PHP can be used on multiple platforms including Windows and UNIX based systems such as Linux, the operating system or computing system does not affect the development of the simulation tool [22].

Reduction in Server Resources required for some operations and Reduced Pricing

Because the PHP simulation engine results in a thinner application, the server processing power and memory requirements are less, leading also to reduced expenditure on the server and cloud computing resources. PHP was designed to be run on open source web servers and platforms, which means any application developed using PHP is well suited for students, schools and SMEs with budget limitations. Furthermore, power systems studies will benefit from having more free alternatives to the premium packages that dominate the market which usually have limits on network size.

Suitability for Smart Networks

With the introduction and proliferation of smart networks [81] and devices that use TCP/IP and HTTP communications for power systems applications [82][83], PHP which is a web programming language and is designed for use over such networks may provide more applications for consumer interaction with future electricity networks.

Database Support

One of the strongest points of PHP is its support for a wide range of databases. By using database specific extensions, e.g. for a MySQL database or MSSQL, or using an abstraction layer like its native PHP Data Objects (PDO), PHP can use data from various sources [22]. The implication is that legacy data from existing systems can be connected to such applications with a level of ease that is not available with other languages, and without modifying that data format.

Interconnectivity

PHP has support for talking to other services using protocols such as LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (on Windows), etc. [22]. Raw network sockets can also be opened to interact using any other protocol. PHP also has support for instantiation of Java objects and using them transparently as PHP objects [22]. This implies that applications created using other languages can be extended to be used in the power systems analysis based on the PHP platform. PHP can also execute shell commands, meaning that a fall back system to a 3-tier system is also possible when the application requires additional functions.

Portability

PHP can be compiled into C++ using the Hip-hop PHP compiler (HpHpC) [84], [85]. The performance gains of C++ are combined with the rapid development paradigm of PHP using this approach. Hip-hop for PHP was developed by the creators of Facebook™ who originally used PHP to develop the popular social networking site but required some of the performance and scalability features of C++ without converting their entire codebase. The Hip-hop Virtual Machine (HHVM) is also available for use to scale PHP applications without compiling them, working like a Just-In-Time (JiT) compiler.

Results presentation

Using PHP allows simulation results to be published in a web-based, interactive manner on the internet or a local network easier than other programs such as PowerFactory or MATLAB. PHP code can easily be embedded in Web Pages, making the integration process easier for results presentation. By using JavaScript, CSS and HTML tools, the otherwise static results may also be given aesthetic modifications unmatched by other platforms.

Simplicity and Learning curve

PHP was designed to be easy to learn and to allow beginners and less experienced programmers to build dynamic websites and achieve complex tasks easily [22], [86]. By using PHP for power systems analysis, engineers with little programming experience can perform analysis and computation without the assistance of third parties and exert full control on their work. Experienced programmers in other C-style languages in general also find it easy to learn.

Availability of Analytical Tools

PHP provides adequate tools for power systems analysis, such as matrix manipulation, complex number analysis etc. The tools for Mathematical computation are provided both natively and via Math extensions [87]. Although there are packages and languages that provide more power and control to carry out functions required for the analysis of power systems, the OOP concepts available in PHP allow the creation of specific functions as required.

5.2.2 Disadvantages of PHP for power systems analysis

The disadvantages and challenges that are likely to be encountered when using PHP for power systems analysis related to this work are as follows:

Language flaws

The flaws of the PHP programming language itself are well documented [79], [86] with issues such as consistency, loose variable types and declarations, programming style, repetitive or obscure function names, scope, comparison operators, etc. Most of the flaws in the language are as a result of the flaws in the original versions with updates retaining some of them for compatibility reasons. This may prove to be a pitfall when writing power systems applications in PHP.

Availability of Programming tools and Libraries for Power Systems Analysis

Languages like C++ for Power systems analysis provide tools that make it easier to perform certain tasks, such as in C++ where native Templates can be used to declare the System admittance matrices [88]. This isn’t the case in PHP and an extension [87] has to be used for matrix operations, or Matrix operations functions must be created by the developer. The same issues are expected for carrying out other forms of mathematical analysis, and it may require significant additional work writing functions specifically to perform these or the use of third party extensions. Furthermore, as no previous power simulation systems have been written in PHP, there aren’t any reusable components apart from those developed in the simulator described in this chapter. It means that any other engineers or researchers will have to use these or lose time re-developing operational components which are already available in packages like MATLAB [74]. This will have to be weighed against the advantages it provides in making a decision to use PHP.

Performance

Since PHP is interpreted and not compiled, PHP programs must be parsed, interpreted, and executed each time each time they run and are therefore usually slower than compiled languages [84]. However virtual machines such as HHVM [85] described previously as well as fast web server applications such as NginX [89] mitigate this interpreted language inherent performance issue significantly.

A lot of the disadvantages and advantages of PHP may be viewed from a more philosophical standpoint, regarding the perceived quality or maturity of PHP as a programming language, or even if it is appropriate to refer to it as one rather than as a scripting language.

The PHP-based solution meets the requirements for a power systems simulation outlined in [15]. It exploits the advantages presented and can produce valid, consistent results comparable with benchmarks from other recognised packages therefore it is a suitable candidate for the investigations.

5.3 2-tier framework for web-based simulation using PHP

An approach which has not been used previously is to develop a slimmer 2-tiered structure for the simulation application as illustrated in Figure 5.1. Instead of using the web server only as a transport layer, a 2-tiered approach which has the simulation engine written in a web programming language and all or most of the processing carried out on the web server is proposed.

Web servers typically run scripts written in ASP, ASP.NET, JSP, Perl, PHP, etc. [28] which are designed primarily for text-based communication using Hypertext Transfer Protocols (HTTP) while simulation packages are built using more powerful languages such as C, C++, C#, Java, etc. [24], [66] because they have a wider range of tools to perform the mathematical analysis required and produce results in a relatively short amount of time. However more recent versions of the web server languages have been upgraded to provide tools for more complex computation such as Object Oriented Programming [26].

PHP: Hypertext Pre-processor, popularly referred to in the recursive acronym PHP, [22] is the most used web server scripting language [90] but WBPSA is currently not being implemented using it. There are several possible reasons for this including the ease of interfacing existing simulation software using ASP.NET to VB and JSP to Java, and the computational limitations of PHP as a programming language for modelling and simulation.

PHP has gradually transformed into a general purpose programming language [22], [26] as web applications have become more complex and the language has been updated. Recent versions of PHP include additional features to make it easier to build complex web applications using PHP. Apart from the additional core features that make PHP more capable of general purpose programming, it also benefits from having several extensions and third party libraries to further extend its mathematical and analytical capabilities.

The implication is that some of the power systems analysis computation that has been handled completely by the dedicated simulation tier may now be performed on the web server tier resulting in a slimmer architecture which is easier to deploy in websites and requires less computational resource requirements.

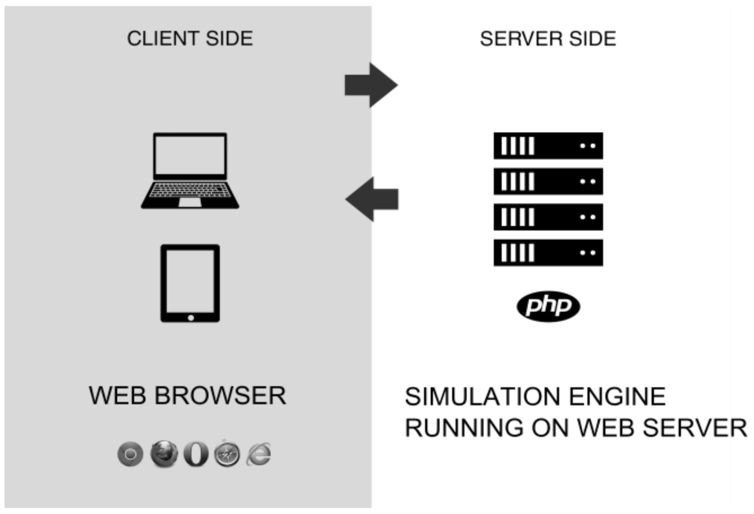


Figure 5.1. 2-tier architecture for web-based power systems analysis using PHP

The approach proposed in this methodology is a modular object-oriented framework in which the major functions required for simulation exist in PHP files as separate models and the elements that make up an electrical network are defined as separate classes with methods and properties.

The components interact with each other via a Controller class that calls the required models upon request and creates objects in memory as the application is executed and then displays the results using a View class for presentation. Functions can be re-used or modified independently of the entire program. Classes are PHP Objects that contain Methods (actions, functions) and Properties (descriptions of physical attributes) for each of the system components [26].

The block diagram in Figure 5.2 show the different modules used in the library, their functions, and the interactions between the modules.

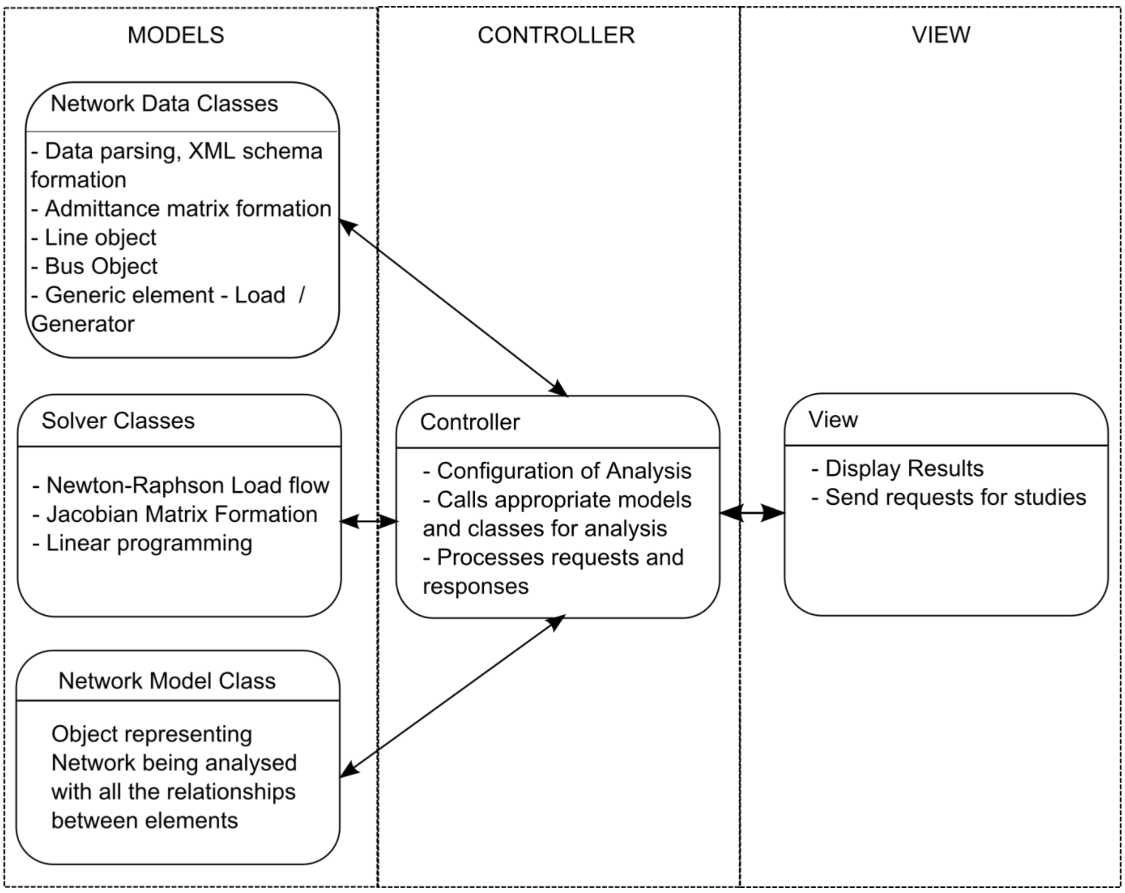

MODELS
CONTROLLER
VIEW
Network Data Classes
- Data parsing, XML schema formation
- Admittance matrix formation
- Line object
- Bus Object
- Generic element - Load / Generator
Solver Classes
- Newton-Raphson Load flow
- Jacobian Matrix Formation
- Linear programming
Network Model Class
Object representing Network being analysed with all the relationships between elements
Controller
- Configuration of Analysis
- Calls appropriate models and classes for analysis
- Processes requests and responses
View
- Display Results
- Send requests for studies

Figure 5.2. Module functions and interactions for PHP WBPSA library

5.4 Extending the functionality of PHP to handle Power Systems Simulation concepts

The most important part of the methodology is how PHP will be extended to handle operations that are not native to it. For power systems computations matrix, complex number and vector operations are the most important to be considered. The PHP core comes with a library for general and basic mathematical operations [91]. It does not provide full native support for matrix operations, complex numbers and vectors.

To use such functions one can either install mathematical extensions [92] which will allow the programmer call functions for matrix operations as though they were native to PHP, or the programmer will have to include a PHP class file that contains methods to perform matrix, vector and complex number operations and then refer to that class each time the operation is required. The PHP group currently doesn’t provide an extension for complex number operations, but have a repository for such classes using a PEAR extension. PEAR stands for PHP Extension and Application Repository and is a framework and distribution system for reusable PHP components [93].

5.4.1 Matrix operations

A matrix may be defined in PHP as arrays of arrays, representing rectangular matrices in row major order.

An array is defined in the PHP manual as an ordered map. A map is a type that associates values to keys [94]. It is a programming concept available in most languages.

Defining a matrix as described limits the programmer to storage and retrieval of data in a particular order as well as some other ordered data operations. To allow full matrix operations such as arithmetic, vector multiplication and finding the determinant, the PEAR Math Matrix package is used in developing this simulation engine. These packages are PHP classes with matrix operations, methods and properties.

Using this method is preferred to using the PHP Lapack extension [95] that provides some of these functions for two reasons; Firstly, the Lapack extension serves a specific purpose only - the matrix and linear algebraic operations, while the PEAR extension is installed once and used for several packages with different functions. Secondly, the classes used in the PEAR extension are written purely in PHP and are only additional PHP files that are included with the application. They can therefore be modified to suit the application easily.

The matrix is instantiated as shown in Table 5.1.

Table 5.1. Matrix Initialization in PHP using Math_Matrix class

Mathematical Operation	PHP operation
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$matrixArray = array(array(1, 2), array(3, 4)); $matrix = new Math_Matrix ($matrixArray);

Now the `$matrix` variable is a Matrix Object [1,2; 3,4] and has all the properties and methods of a matrix, rather than being just a multidimensional array which only stores and retrieves the data in an ordered manner.

The limitation of the Math_Matrix class is that it can only accept real numerical values. The obvious problem with this in power systems analysis is that a bus admittance matrix is made up of complex number values. To solve this problem, the complex admittance matrix is organized and manipulated using native PHP multidimensional arrays while other matrices, such as the Jacobian matrix, which consist purely of real values and require other matrix operations are formed using this matrix class.

5.4.2 Complex Number and Vector Operations

PHP doesn’t natively handle complex number operations such as conjugation, inversion and determining angles which are required for power systems analysis. The PEAR Complex number package [96] is used to enable this functionality. Some operations such as solution of linear algebraic equations involving Jacobian matrices require vector analysis; the vector in this case referring to a one-dimensional array of real and reactive power.

The matrix package for example solves linear equations using an iterative error correction algorithm and requires as input the left hand side vector and a right hand side matrix. To enable this type of functionality the PEAR vector package [97] is used in this methodology as it allows the use of vector methods and properties on variables in the simulation engine.

Table 5.2. Complex number and Vector definition in PHP

Mathematical Operation	PHP operation
Complex number: $1 + 2i$	$complex_number = new Math_Complex(1,2);
Tuple (vector): $(1, 2, 3, 4, 5)$	$vector = new Math_Vector(1,2,3,4,5);

5.5 Electrical Network definition in PHP

The network data is defined using an XML format document which is parsed and read into memory by the PHP library. This is suitable because of the flexibility of this format compared to the rigid, tabular structure of relational databases such as MySQL and Oracle. Flexibility in the data structure is important as different elements may be present or absent at nodes in electrical networks and it will be more difficult to represent this in tabular relational databases. The network topology in some cases may be complex leading to complexity in a relational structure. Elements may also have parameters that do not exist in other elements therefore having an elements table for example will be difficult to implement without flexibility.

The network is defined using different custom tags for the elements, the buses and lines. Loads and generators are defined based on the direction of power flow. The network shown in Figure 5.3 is defined by the XML code in Listing 5.1.

Listing 5.1. Sample 3-bus Network Definition in XML

```
<?xml version="1.0" encoding="UTF-8"?>

<PowerNetwork    realPowerUnit="MW"    reactivePowerUnit="MVAR"
apparentPowerUnit="MVA">

    <NetworkBuses>

        <Bus number="1" voltagePU="1.05" vAngle="0"></Bus>

        <Bus number="2">

            <Element p="4" flow="out" />

            <Element q="2.5" flow="out" />

        </Bus>

        <Bus number="3" voltagePU="1.04">

            <Element p="2.0" flow="in" />

        </Bus>

    </NetworkBuses>

    <NetworkLines>

        <Line from="1" to="2" zReal="0.02" zIm="0.04" />

        <Line from="1" to="3" zReal="0.01" zIm="0.03" />

        <Line from="2" to="3" zReal="0.0125" zIm="0.025" />

    </NetworkLines>

</PowerNetwork>
```

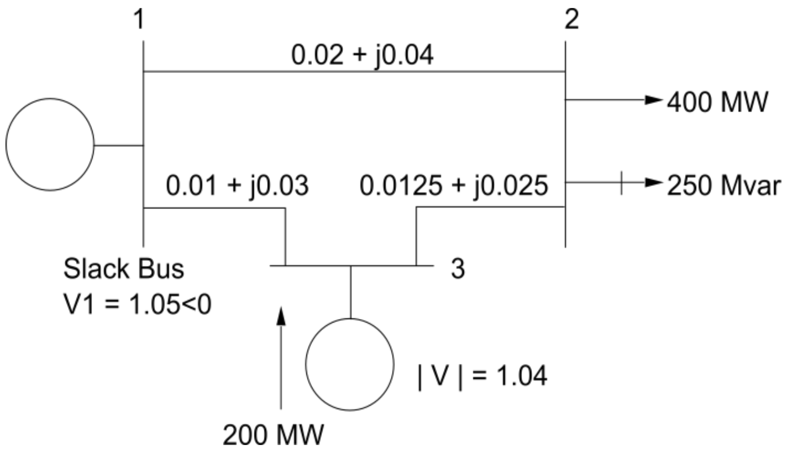


Figure 5.3. Sample 3-bus network to be defined in XML

The classes described in Table 5.3 have been developed to form an electrical network in the PHP application which is ready for any operation or computation such as power flow or short circuit analysis and using any chosen method. With the exception of the *jacobianMatrix* property in the *Network* element, these are the bare minimum required objects for any other power system analysis that may be carried out. This structure means these classes can be easily re-used in other PHP applications.

Table 5.3. Electrical network definition in PHP power systems analysis library

s/n	**Class Description**	**Methods**	**Properties**
1	**Network Class:** Represents an electrical network with no elements. It is the main dependency for all other classes as instances of the element class are added to this class to build the network.	**InitializeNetwork:** sort buses and lines in a particular order, set slack bus, compute bus unknowns and call method to form admittance matrix for this network **addBus**: add a bus object to this network **addLine:** add a line object to this network **Bus**: returns a particular bus object in the network by number **BusRealPower, BusReactivePower:** returns the real and reactive power of a given bus **BusUnkowns:** compute unknown parameters for given bus **formAdmittanceMatrix**: Forms the network admittance matrix from the lines and respective buses given **delPdelQMatrix**: returns a vector of the change in power for the linearized relationship involving change in power, voltage magnitude	**Buses:** array of bus objects in the network **Lines:** array of line objects in the network **admittanceMatrix:** Multidimensional array of line admittances **jacobianMatrix:** Matrix object representing Jacobian matrix from latest iteration **slackBus:** integer of slack bus number; default is 1 **voltageControlledBuses:** Array of voltage controlled buses; filled during network initialization **initialV:** initial voltage magnitude for iterations, default is 1.0 per unit; **initialD:** intial voltage angle for iterations, default is 0; **solution:** array for storing power flow solution

		and angle and the Jacobian matrix	**dPdQM_network:** vector of change in real and reactive power
2	**Bus Class:** Object representing a power bus, or a node in an electrical network with all its properties and methods	**__construct:** set bus number, specified voltage magnitude PU and angle both default to null **addElement:** Adds an element object to this bus and update the properties of this bus according to the properties of that element object, whether it is a generator or a load.	**P:** Bus real power; **Q:** Bus reactive power **S:** Bus apparent power; **Type:** Slack, PQ or PV **Elements:** Array of elements at this bus; **Unknowns:** array of bus unknowns filled during network initialization; **previousV, previousD, previousP, previousQ:** variable to store previous values of voltage magnitude, angle and power during iterations
3	**Line Object:** represents a line in an electrical network	**__construct:** initialize the line, set the numbers for connected buses and value for line impedance real and imaginary parts	**From:** number of first connected bus **To:** number of second connected bus **Impedance:** complex number object with impedance value **Admittance:** complex number with admittance value **Label:** string to label this

			line in from-to notation.
4	**Element Object:** Object representing a generic element in the network. The element may either be a generator or other active element or a load.	**__construct:** initialize element and set its properties. The properties default to null so they may be set after initialisation.	**S:** a complex number object representing the apparent power. Computed from the values of real and reactive powers given or declared explicitly. Can be negative or positive depending on the *flow* property. **Flow:** String representing direction of power flow in or out of the network in relation to this element **P:** Real power **Q:** Reactive power **Name:** user defined name for the element

5.6 System architecture of PHP library for power systems analysis

The system architecture for the library is shown in the block diagram in Figure 5.4 which illustrates the dependencies and interrelationships of all components of the library in a block diagram. The dependencies for any study to be carried out are included in the program before the operations are run. After the network model is instantiated any power systems study may be performed on it.

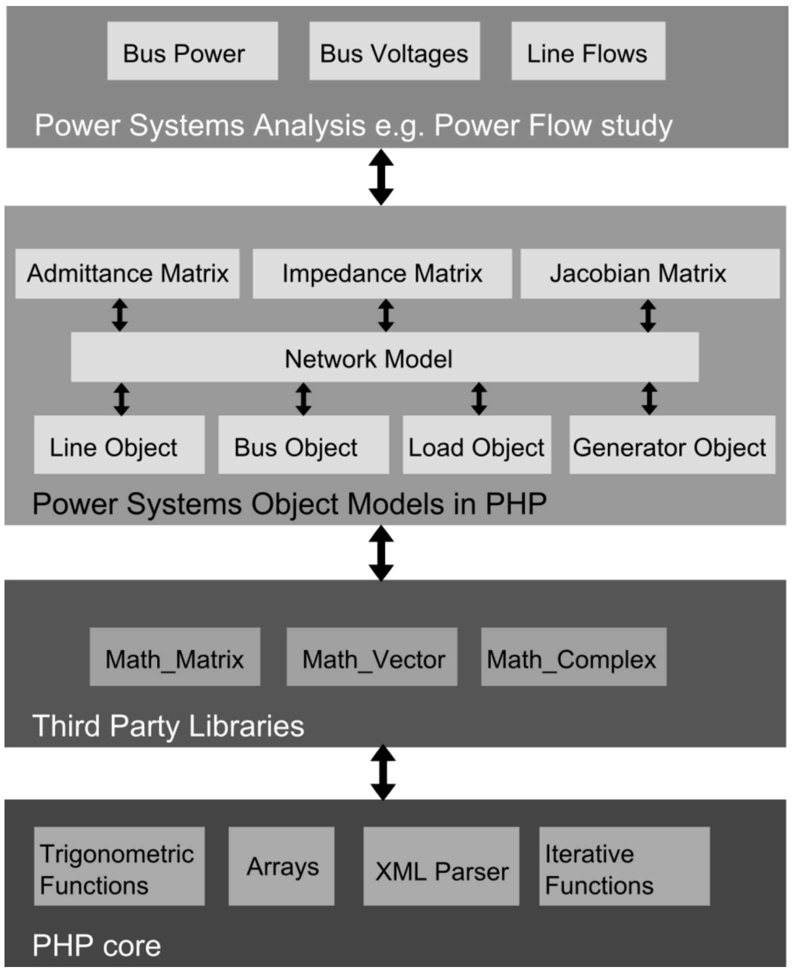


Figure 5.4. System Architecture for PHP WBPSA library

5.7 Power flow solution in PHP using classical Newton-Raphson method

To perform power flow studies in the PHP library a number of classes are built to interact with the network models described in the preceding sub-section. The Newton-Raphson method is one of the popular and dependable methods for carrying out a power flow study and uses a Jacobian matrix to obtain linearized relationships between parameters and therefore generate a solution via iteration. Table 5.4 describes the classes and methods used to perform the Newton-Raphson power flow in the PHP library.

The Classes described in Table 5.4 are used to perform a power flow using the Newton-Raphson (NR) method on a valid PHP electrical network object. The modular structure of this methodology means that any other solution may be used alongside the NR on the same network and won’t require the entire application including the network definition classes to be developed again.

Table 5.4. PHP Classes for Newton-Raphson power flow solution

	Class Description	Methods	Properties
1	**JacobianMatrix:** Contains the methods for the formation of the Jacobian matrix for a given network	**delPdelD, delPdelV, delQdelD, delQdelV:** These methods form the sub matrices of the Jacobian Matrix **FormMatrix:** takes a power network object and runs the methods for the Jacobian sub-matrices on it and returns a full Jacobian matrix as a matrix object. This object is stored in the Network's *JacobianMatrix* variable.	No properties declared as all elements can be accessed from matrix object returned on its formation.
2	**lfNR (Load Flow Newton Raphson):** Computes a Newton Raphson power flow solution for a given network according to the NR algorithm.	**exec:** this executes the individual steps of the power flow by forming the Jacobian Matrix, forming the power change vector and using the Matrix Object linear equation solution to obtain a step solution. **updateNetwork:** updates the buses of the network with the latest results from an Iteration step **solve:** Iteration function that calls the exec function at each step and checks the tolerance of the solution before finally computing	**maxIterations:** maximum number of Iterations, to prevent infinite iterations **e:** variable for acceptable convergence criteria **step:** current iteration step

		slack bus values when given convergence criteria is met or set number of steps exhausted	

The PHP library Power flow solution using the Newton-Raphson follows the same flow chart structure as a generic power flow using any other method described in [18]. The specific flowchart for this PHP application and methodology is shown in Figure 5.5 and describes the interaction between the classes.

This same structure can be incorporated into other power systems studies that require a power flow analysis to be carried out as a step, such as optimal power flow computation. The full process may also be broken into smaller processes because of the modular nature of the application.

The flowchart shown in Figure 5.5 is separated into two parts; the first part shows the process of creating a power network from the network data provided and is the basic requirement for the simulation. This network resides in the PHP application and can be used for any other analysis such as short circuit studies, reliability analysis, etc. The second part of the flowchart shows the process for the power flow carried out on the network, and can be replaced by another process.

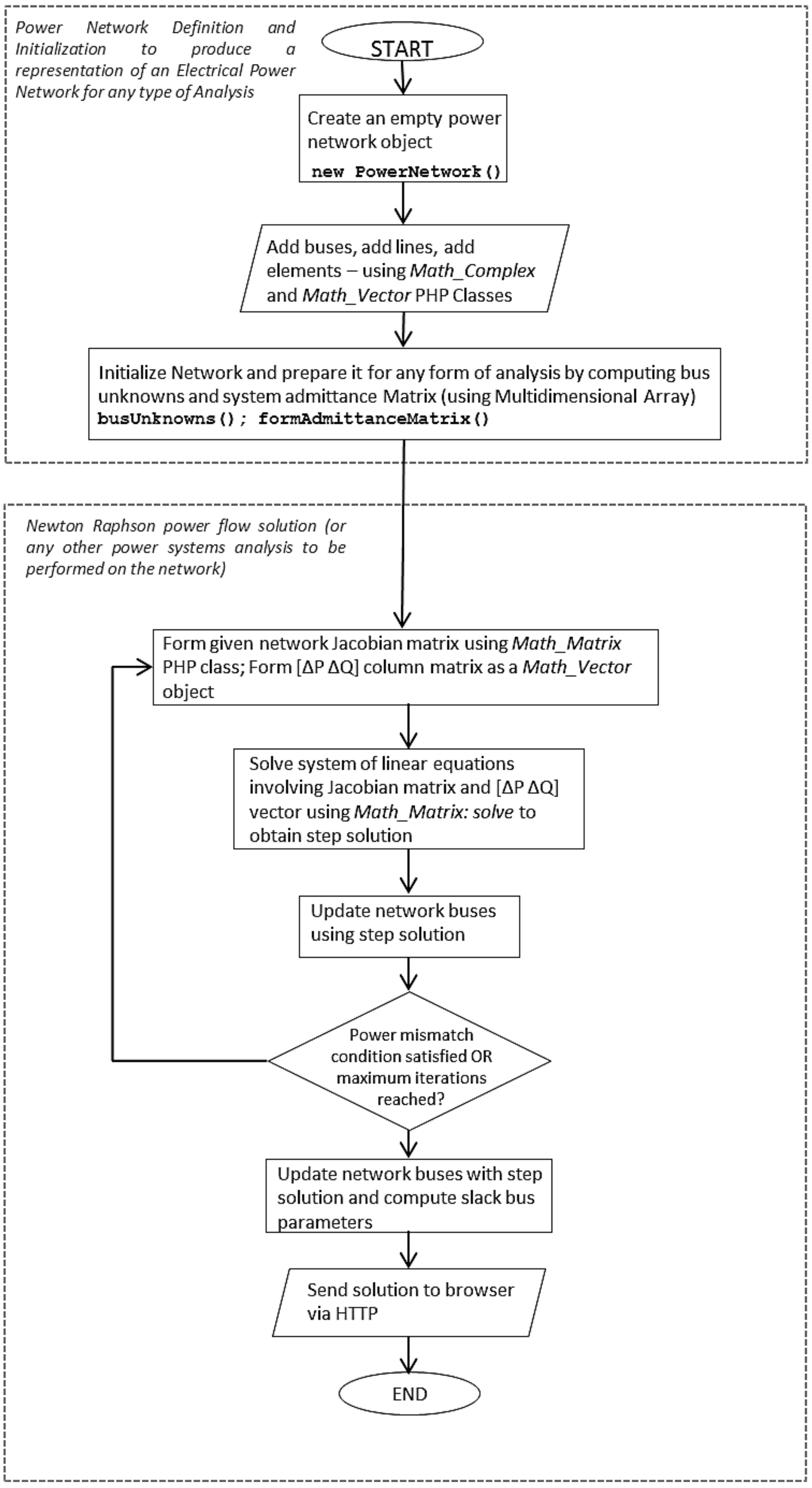

Power Network Definition and Initialization to produce a representation of an Electrical Power Network for any type of Analysis
START
Create an empty power network object
new PowerNetwork()
Add buses, add lines, add elements – using Math_Complex and Math_Vector PHP Classes
Initialize Network and prepare it for any form of analysis by computing bus unknowns and system admittance Matrix (using Multidimensional Array)
busUnknowns(); formAdmittanceMatrix()
Newton Raphson power flow solution (or any other power systems analysis to be performed on the network)
Form given network Jacobian matrix using Math_Matrix PHP class; Form [ΔP ΔQ] column matrix as a Math_Vector object
Solve system of linear equations involving Jacobian matrix and [ΔP ΔQ] vector using Math_Matrix: solve to obtain step solution
Update network buses using step solution
Power mismatch condition satisfied OR maximum iterations reached?
Update network buses with step solution and compute slack bus parameters
Send solution to browser via HTTP
END

Figure 5.5. Flowchart for Newton-Raphson Power Flow study in PHP WBPSA library

5.8 Balanced Symmetrical 3-phase fault computation methodology

The short circuit fault study implemented in this methodology is the balanced symmetrical 3-phase fault which has the same current flowing through all phases. Table 5.5 describes the classes and methods used to perform the fault study in the PHP library. The class depends on the output of the other classes i.e. the Power Flow class.

Table 5.5. PHP Classes for balanced 3-phase short circuit fault study

	Class Description	**Methods**	**Properties**
1	**shortCkt:** Contains the methods for performing a short circuit study for balanced 3-phase faults	**Initialize:** performs a power flow study on the network to get base case conditions and admittance matrix and also set faulted bus number **formImpedanceMatrix:** Takes the admittance matrix and forms an impedance matrix using the inverse of the matrix. **faultCurrent:** compute fault currents using self-impedance element of impedance matrix and initial voltages **faultVoltages:** computes fault voltages in all buses using result of **faultCurrent** and impedance matrix entries by looping through buses with values of mutual impedances **shortCktCapacity:** Computes short circuit capacity of faulted bus in MVA using base MVA and	No properties declared as all elements can be accessed from matrix object returned on its formation.

		per unit reactance at faulted node	

The flowchart in Figure 5.6 shows the process for performing the analysis. The entire functionality of the power flow solution displayed in Figure 5.5 is collapsed into a single block however the common functionality is displayed in the top of the flowchart. The flowchart is more linear than the power flow study chart because the functions do not require as much iteration and conditional statements. The only input is the faulted bus number, and the calculations for the other parameters are based on the results from the calculations on this bus and the output of the impedance matrix formation.

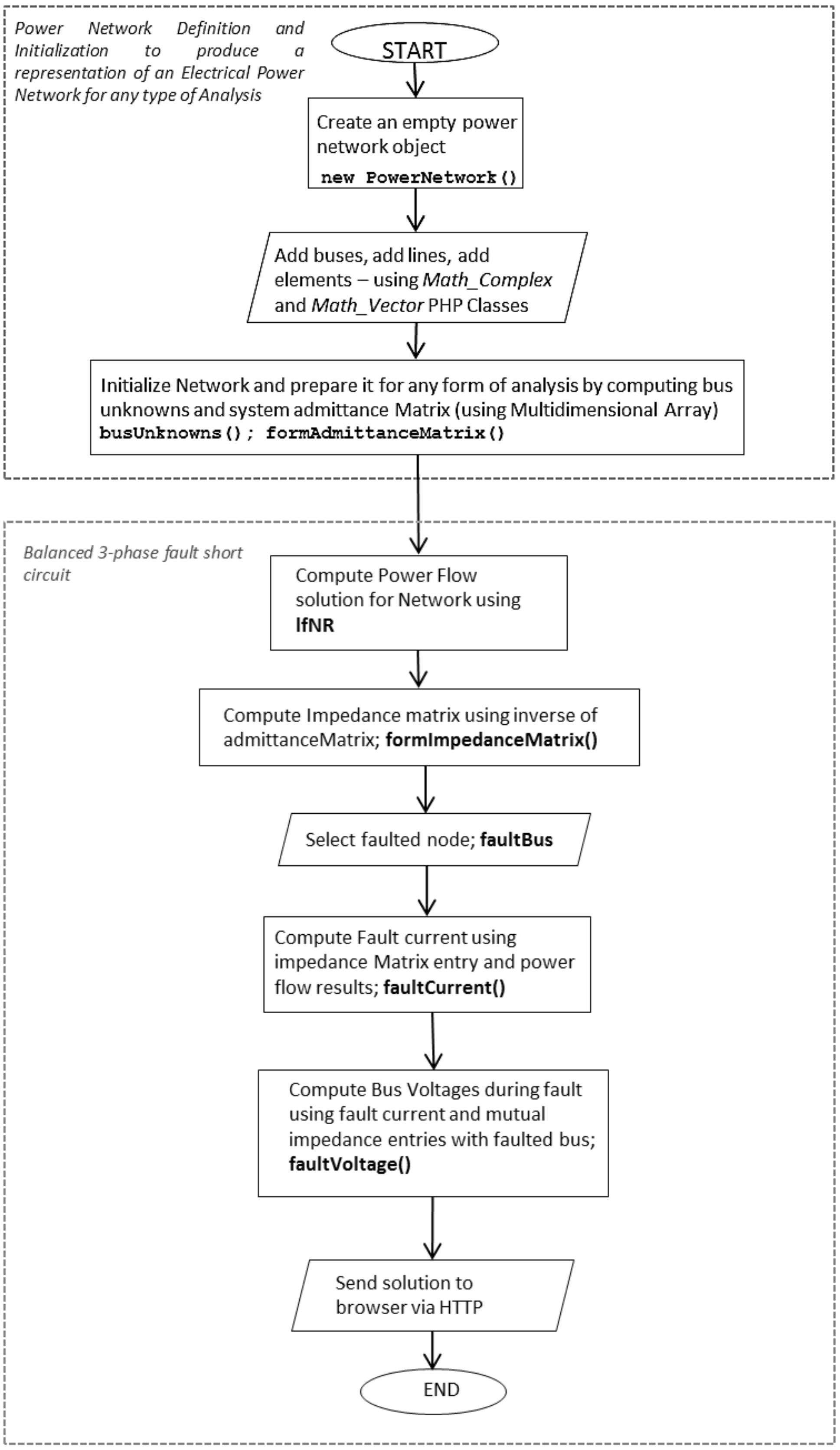

Power Network Definition and Initialization to produce a representation of an Electrical Power Network for any type of Analysis
START
Create an empty power network object
new PowerNetwork()
Add buses, add lines, add elements – using Math_Complex and Math_Vector PHP Classes
Initialize Network and prepare it for any form of analysis by computing bus unknowns and system admittance Matrix (using Multidimensional Array)
busUnknowns(); formAdmittanceMatrix()
Balanced 3-phase fault short circuit
Compute Power Flow solution for Network using lfNR
Compute Impedance matrix using inverse of admittanceMatrix; formImpedanceMatrix()
Select faulted node; faultBus
Compute Fault current using impedance Matrix entry and power flow results; faultCurrent()
Compute Bus Voltages during fault using fault current and mutual impedance entries with faulted bus; faultVoltage()
Send solution to browser via HTTP
END

Figure 5.6. Balanced short circuit study in PHP flowchart

6 Evolutionary combinatorial optimisation for Energy storage scheduling: Case Studies, Results and Discussion

6.1 Introduction

This section presents the results obtained from testing the evolutionary combinatorial optimisation algorithm on four case studies. Each case study will include details on the data sources, the method of collection, and a background on the circumstances surrounding the data to provide context. All data used for the tests is provided explicitly for clarity and for readers who wish to reproduce the results or compare with other methods.

In all test cases the algorithm is off-line, meaning that it is assumed to have perfect foresight of the future demand characteristics from a forecasting model.

The case studies aim to verify the feasibility of the algorithm and evaluate it in the following terms:

The impact of using the algorithm on a network demand profile previously without energy storage to verify if it creates new problems or resolves peak demand and load levelling issues

A quantitative comparison of the performance of the algorithm by measuring its results against results obtained after other verified scheduling methods are applied to the same input data to determine if results are within the same range and if there are any correlations between the results. This comparative analysis methodology is illustrated in Figure 6.1.

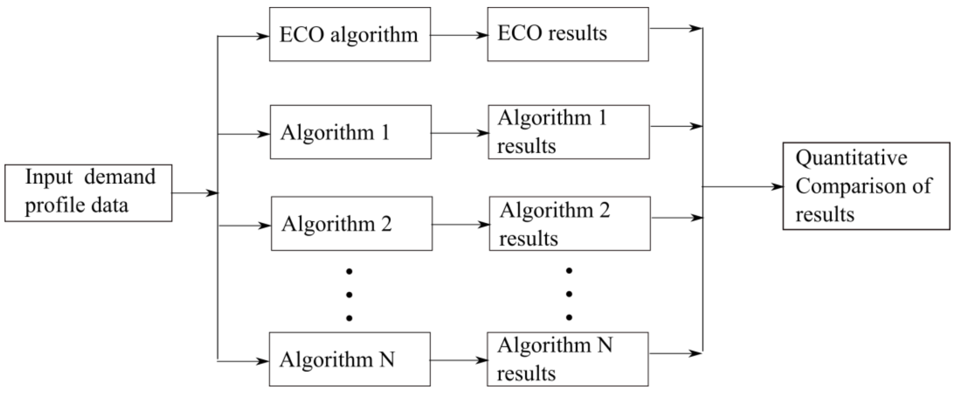


Figure 6.1. Quantitative comparative analysis methodology

The simulation setup is the same in each case; The algorithm is implemented in JavaScript on a web server using the NodeJS runtime [98] with results displayed in the command line console and a web browser. The NodeJS JavaScript runtime is used because of the quick computation speed and non-blocking input-output (I/O) characteristics that allow results to be returned asynchronously for multiple calculations. This is ideal for heuristic calculations which have different results that can be compared. Other languages such as Python, C++ or MATLAB may also be used to implement the algorithm according to requirements. Each time the algorithm is run a viable and usually different solution is obtained because of the heuristic nature of the algorithm and the random selection of combinations for the subset sum. Asynchronous returning of results means different algorithm result schedules can be sent as output to other processes such as comparison processes while other viable schedules are being generated. The program does not have to wait for all schedules to be generated before checking which has better peak reduction or load-levelling. The benefit is an improvement in performance as generation and comparison processes can run concurrently.

6.2 Case Study: Riverside Community, Stirling, Scotland UK

6.2.1 Background

Riverside Community is located in Scotland UK, in the heart of Stirling down town at latitude 51°10' N and longitude 3°45' W. The area is dominantly domestic sector with a small percentage of commercials [99][100].

As a part of a scheme by the Riverside community and Stirling council to become a carbon neutral community, i.e. a community with a net zero carbon footprint achieved by reducing their carbon emission by following a set of practices, the council commissioned a study in 2006 - 2007 in collaboration with the University of Strathclyde relating to the energy demand of the council. The objectives of the study include assessing the energy demand and carbon emission in Riverside community and to assess possible demand and carbon reduction options.

This case study explores using a Battery energy storage system (BESS) to perform peak demand reduction and load levelling on the aggregated electricity demand of the community. If the BESS is dispatched efficiently it will not reduce or increase the amount of energy used by the community, but will be able to shift its own energy capacity across the time period i.e. the same amount energy will be used but at different times. The benefit of this to carbon emissions is that it reduces the need to use peaking generators to supply peak loads, as they can be supplied with a base load generator when the daily demand is lower and levelled.

The objective of this case study is to test the impact of the Combinatorial Optimisation algorithm on the demand profile using different ESS energy block configurations and focusing on the Bin-Packing and Subset sum algorithm aspects (BP-SS).

6.2.2 Test Data and Source

The data used for the case study was obtained from the project carried out by the Stirling Council and University of Strathclyde for Riverside community [99]. The demand of residents was measured using a detailed survey conducted in the community and a modelling tool that takes into account the household stock, building stock, occupancy levels, and consumer behaviour. It is based on probability models that predict the possibility of each household to operate a certain amount of appliances on a certain time of the day for different end users, i.e. occupancy types. The database was compiled with respect to the National Statistical data for electricity used in appliances in UK households, usage patterns, and a survey of residents. It was found from a site survey that as at 2007 the community had 1,015 households with different occupancy levels as shown in Table A.1 in the appendix.

The average peak daily demand for the entire community in winter obtained is 1,019 kW per day between 19:00 and 20:00 and the maximum energy demand is 23,979 kWh per day. It is assumed that the community is connected to the distribution grid at the grid connection point (GCP) substation via a radial network, i.e. the entire community is served from one feeder before further distribution takes place to the households. The daily consumption by household occupancy type is shown in Table A.2 in the appendix.

A 500 kW / 2000 kWh grid scale battery such as [101] with an efficiency of 0.77 was selected to be used for this study and is assumed to be connected and operated at the GCP. The ESS rating and capacity represents approximately 50% of the daily peak demand and slightly less than 10% of the daily energy demand to be shifted.

Before the algorithm was applied a pre-processing operation was carried out to scale the values of data such that it is compatible for use in the algorithm. The power is first converted to per unit on a 100 kW base so the peak becomes 10.19 per unit (pu) and then each demand value is scaled to a maximum of 10pu. The peak battery power becomes 5 pu and its capacity becomes 20 h pu. An energy block of 1 “unit” represents power of 1 pu charged or discharged for a period of 1 hour therefore a 1 unit block is proportional 1 pu of demand or supply for 1 hour and equal to 1 h pu. This means a maximum of five 1h pu unit blocks may be placed in one hour without exceeding the rated power of the ESS. The efficiency is applied to the energy blocks during the charge and discharge phases before they are placed in each time period on the demand profile.

Table A.3 shows the hourly demands after the scaling operation and this is plotted in Figure 6.2. As a requirement of the combinatorial optimisation algorithm the original demand profile shown in Figure 6.2 is used for the charging phase and the inverted demand profile after the mirror transform in Figure 6.3.

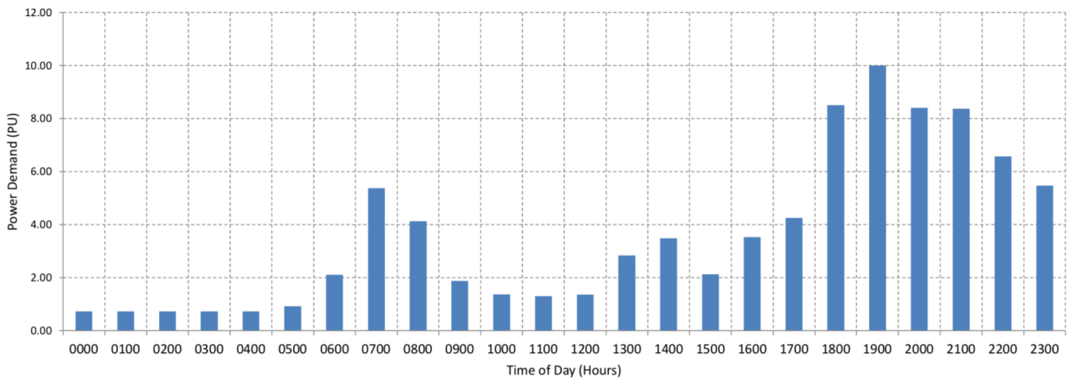


Figure 6.2. Original Riverside demand profile after conversion to PU and scaling

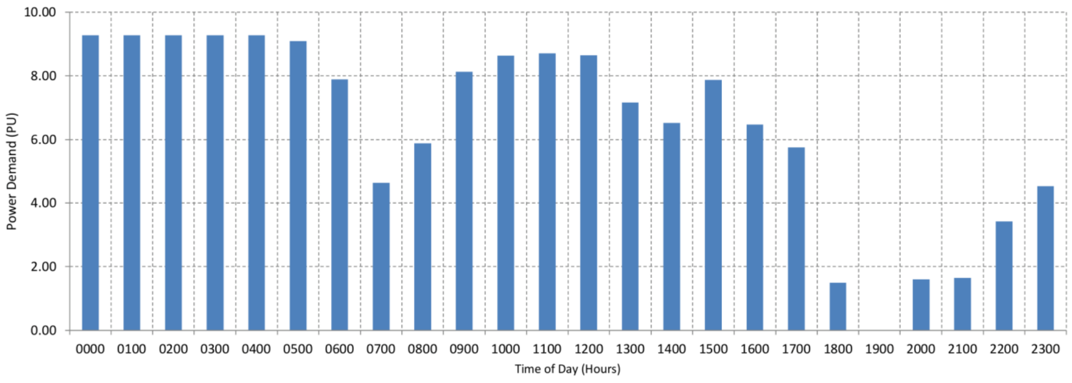


Figure 6.3. Inverted Riverside demand profile after mirror transform for discharging

6.2.3 Results and Discussion

Three separate tests were carried out to establish the changes in result and performance as the number of ESS energy blocks generated from the subset sum algorithm changed. A new set of blocks is generated from the subset sum generator as a result of the random number generator. The Bin-packing algorithm was run on each set of generated blocks several times and the representative average results are presented. The Genetic Algorithm optimisation was not used in this case. The number of blocks used were 6 energy blocks in scenario I (s.I), 11 energy blocks in scenario II (s.II) and 20 energy blocks in scenario III (s.III), with each set summing up to the total capacity of 20 h pu.

After running the algorithm in the different configurations, the charging and discharging schedules of each configuration is shown in Figure 6.4 – Figure 6.6. The summary of numerical results of all the scenarios is shown in Table 6.1 which indicates the amount of peak reduction observed and the peak-to-trough margin which is an indicator of the amount of load levelling. The plots of the final profiles of all scenarios compared to the original profile are shown in Figure 6.7.

Table 6.1. Results of Combinatorial Optimisation algorithm for scheduling ESS in Riverside community

Parameter	**Scenario I: 6 energy blocks**	**Scenario II: 11 energy blocks**	**Scenario III: 20 energy blocks**
Original peak (pu)	10 at 1900	10 at 1900	10 at 1900
Final peak (pu)	8.51 (at 1800)	8.51 (at 1800)	6.15 (at 1900)
Peak reduction amount (pu)	1.49	1.49	3.85
Peak and trough demand margin (pu)	7.66	7.06	4.66
Original number of distinct power demands (nearest integer)	10	10	10
Final number of distinct power demands (nearest integer)	7	7	6
Energy block allocation (pu)	1,2,2,5,5,5	1,1,1,1,1,1,2,3,3,3,3	1,1,1,1,1,1 … (x20)
Charging periods	6 (0000 – 0400, 1200)	10 (0000 - 0500, 0900 – 1200)	12 (0000 – 0600, 0900 – 1200, 1500)
Discharging periods	5 (1800 – 2200)	5 (1800 – 2200)	7 (0700, 1800 – 2300)

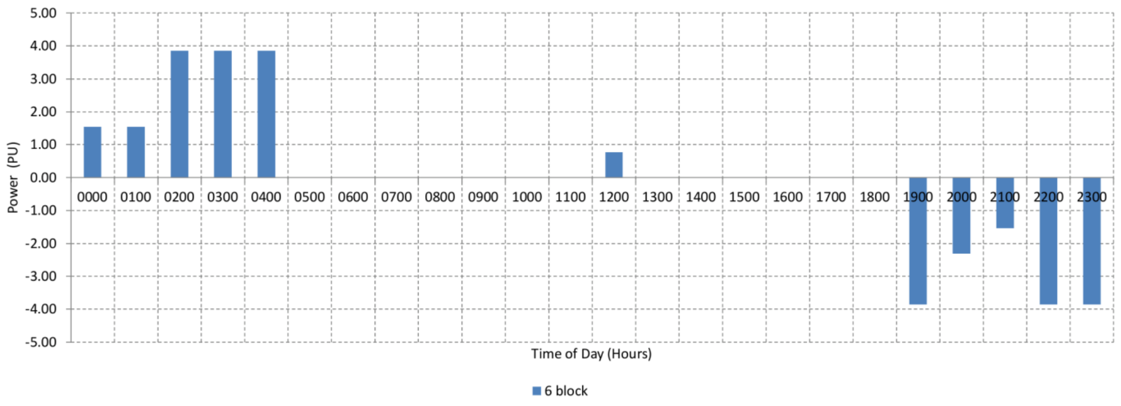


Figure 6.4. 6-block scenario ESS charge and discharge schedule.

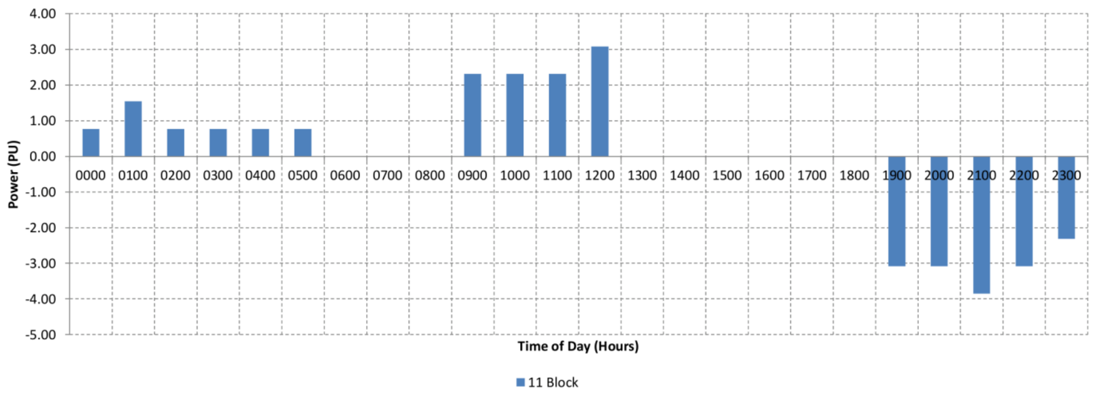


Figure 6.5. 11-block scenario ESS charge and discharge schedule.

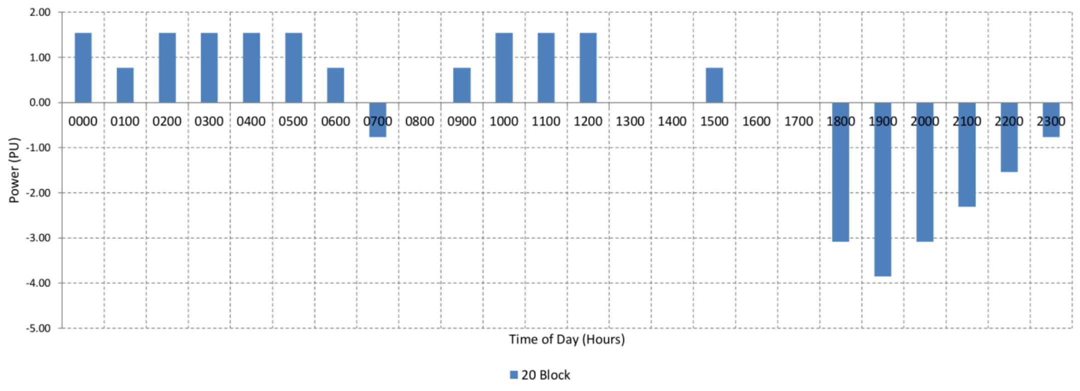


Figure 6.6. 20-block scenario ESS charge and discharge schedule.

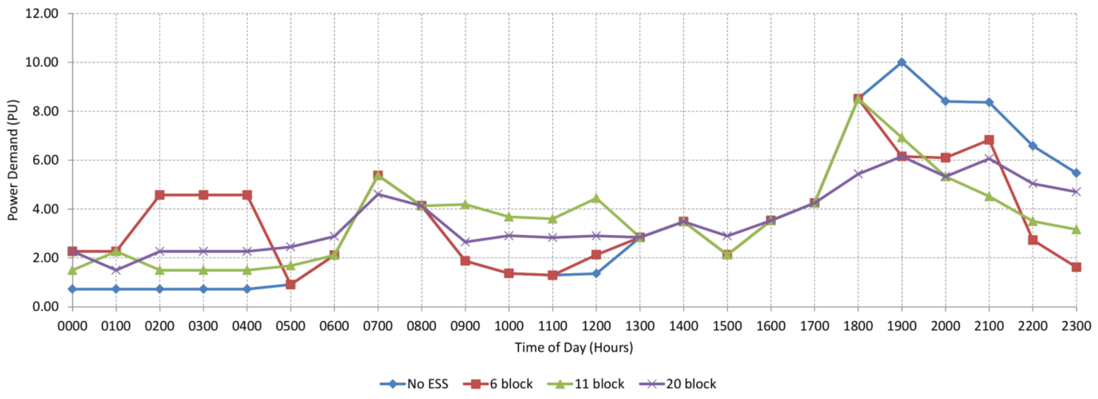


Figure 6.7. Comparison of different scenarios of energy block numbers and normal operation on algorithm completion.

The results show that the amount of peak reduction and the number of peaks tend to increase with the number of blocks. The final lowest demand also increases with the number of energy blocks which indicates levelling of demand.

In s.III for 83% of the day the average difference between the demands is only about 2 pu; the overall difference peak and trough demand is 4.66 pu which indicates a smoother and levelled profile, and the new peak is 6.15 pu indicating a reduction of 3.85 pu. In s.I for 41% of the day the average difference in demand is within 4 pu; the overall difference between peak and trough demand is 7.66 pu. In s.II for 62% of the entire period the average change in demand is within 2 pu; the actual difference between peak and trough is 7.06 pu. Both s.I and s.II resulted in a time-shift in the peak period and a reduction by 1.49 pu to 8.51 pu. In the original profile for 50% of the time the average change in demand is approximately 3 pu; the actual difference between peak and trough demand is approximately 9 pu. This indicates that the 3 scenarios provide a flatter, more predictable demand profile compared to the original profile without the battery. There are also fewer distinct power demands in all 3 scenarios (7, 7 and 6 compared to 10 in the original profile) which shows a reduction in the demand fluctuation.

In the 3 scenarios the energy blocks are all used during the charge and discharge cycle; which is effective battery utilization. Leftover uncharged blocks reduce the amount of demand shifting that can be done. When there are leftover charged blocks after the discharge cycle then the battery would have contributed as a load to the demand.

The subset sum algorithm is essential because of the limited number of charge/discharge cycles for a given battery. When the number of blocks is higher the number of charge cycles (6, 10 and 12 respectively) and discharge cycles (5, 5 and 7 respectively), shown in Figure 6.4 – Figure 6.6 , increases and this affects the usable lifespan of the battery. Also a solution takes longer to obtain with more energy blocks as the algorithm will have more items to pack. The subset sum algorithm is therefore necessary for finding an optimal combination and number of energy blocks for the best results in demand levelling and the lowest number of charge and discharge cycles.

In all scenarios the amount of energy supplied from the GCP with or without the battery remains the same at 90 pu. The battery can shift its total energy capacity around the demand profile during the redistribution if the entire capacity is charged and discharged. The algorithm is verified in this case because the energy transferred from the GCP is the same remains the same with or without the ESS. In summary, the peak demand was reduced by up to 38.5% and the demand margin between peak and trough demand was reduced by up to 50.2% using the 500kW/2000kWh ESS with the combinatorial optimisation algorithm and without further optimisation by genetic algorithm.

6.3 Case Study and algorithm verification comparative analysis: Combinatorial optimisation algorithm compared to set-point control in household demand smoothening

6.3.1 Background

The demand smoothening and peak shaving algorithm (DS) was developed in [36] by Purvins et. al for households to reduce their daily peak demands and variation using a battery energy storage system (BESS). The analysis included case studies of a number of European countries in different seasons and the most data was reported for the tests performed for Denmark in spring.

The method uses an average demand value as a reference point to generate a charge and discharge matrix to inform the final schedule. Two models were developed in the study; a time-dependent model that used a fixed average set-point and a demand tracking model that used a dynamic set-point that changed with the demand forecasted. The results of the demand tracking (DT) method of the DS algorithm showed better performance than the time-dependent model and therefore these results were selected for the comparative analysis to be used for algorithm verification.

A case study and comparative analysis was carried out to verify the combinatorial optimisation (CO) algorithm and measure its performance against a verified algorithm. The objective of this analysis is to observe the output of the algorithm when the same input is used and find correlations which indicate that the algorithm is comparable, competitive and feasible.

6.3.2 Test data and source

The demand profile used for the comparison is a representative average daily demand profile for a household in Denmark in spring time obtained from the Residential Monitoring to Decrease Energy Use and Carbon Emissions in Europe (REMODECE) database [102]. The demand profile and battery parameters were converted to per-unit (PU) values and scaled to a maximum of 10 pu for compatibility with the proposed CO algorithm. The original and scaled data is given in Table A.4 and the demand profile is plotted in Figure 6.8. The required inverted demand profile for discharging is plotted in Figure 6.9.

A battery system of 0.2 pu rated power and 0.4 h pu energy capacity and an efficiency of 0.77 using a base of 0.895 kW is used in [36]. Scaling these values results in a rated power of 2 pu and capacity of 4 h pu.

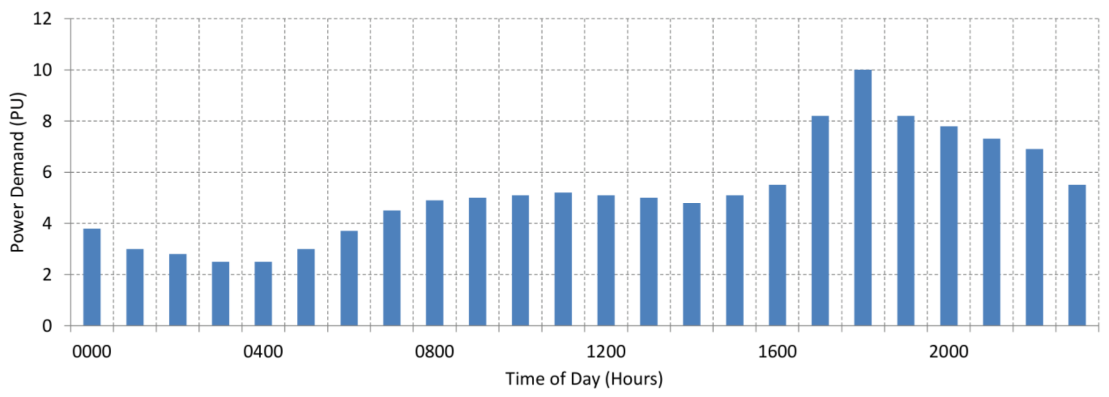


Figure 6.8. Representative average demand profile for household in Denmark in spring

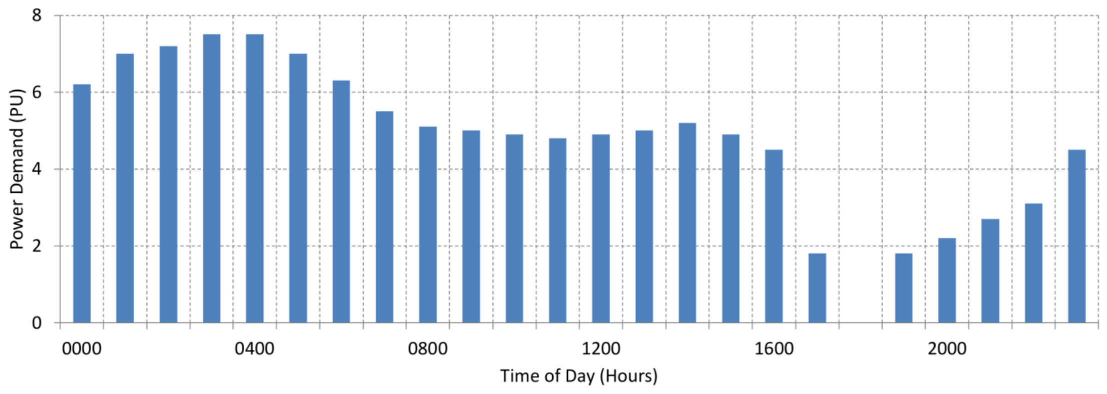


Figure 6.9. Representative average demand profile for household in Denmark in spring after horizontal mirror transform

6.3.3 Results and Discussion

The resulting demand profile obtained after running the CO (bin packing + subset sum) algorithm on the same input is shown in Figure 6.10. The CO algorithm performed better in peak shaving than the DT method, both algorithms shaved the peak demand from 10 pu to 7.69 pu (23.1% reduction) and 8 pu (20% reduction) respectively. The CO algorithm also increased the capacity headroom by reducing ESS demand in the first charging period.

The ESS usage is also less intensive in the BP-SS method as there are 4 charging periods and 2 discharging periods as compared to 8 and 4 in the DT method which represents 50% more intensive use of the battery. The DT method performed better in load-levelling in the first charging period, but they are almost identical in load-levelling for the remaining periods. The DT is susceptible to periodic surges in demand, for example a TV pickup as experienced in the UK [37] which is a disadvantage. The CO algorithm cannot be affected by this because it doesn’t depend on a fixed reference point based on an average. Furthermore, the CO method provides more opportunities for optimisation by varying the number and sizes of energy blocks but further optimisation of the DT method is based only on adjusting the reference average.

The aim of the case study to verify the algorithm using the result ranges has been achieved as it can be seen that the results are within range of verified results and even perform better without further optimisation.

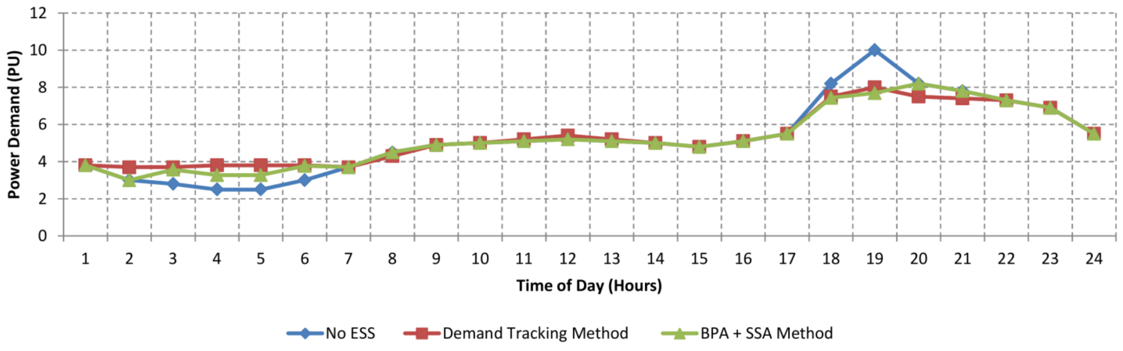


Figure 6.10. Comparative analysis of combinatorial optimisation method and demand tracking set-point

6.4 Case Study and comparative analysis: Leighton Buzzard smart network storage project

6.4.1 Background

The UK power networks (UKPN) is a distribution network operator (DNO) covering London, the South East and East of England [5]. One of the problem areas with regards to network firm capacity violation is Leighton Buzzard (LB), a town in Bedfordshire England located at 51°54′59″N 0°39′42″W, and the UKPN elected to defer traditional network reinforcement opting instead to use ESS to operate the network within limits and ensure security supply among other benefits.

After consultations in July 2013 on possible future business models for grid-scale ESS, in October 2013 the design and planning considerations were released highlighting the future plans for the network [7].

i. This approach of using ESS and this site were selected as LB met criteria including:
ii. High cost, complexity and time consumption of upgrade
iii. Upgrade would lead to addition of potentially unutilised over-capacity
iv. Constraint driving upgrade may be relieved by a relatively small addition of capacity headroom amount

v. Upgrade could be deferred for 5+ years with no forecasted constraints requiring upgrade

The Leighton Buzzard primary substation comprises two 33/11kV 38MVA transformers fed by two 33kV overhead lines with a winter rating of 35.4MVA each. Redundancy is provided to meet the P2/6 engineering requirement [103] for security of supply that sets out a minimum proportion of demand that must be met following the loss of one or more circuits at a site. In this case only the "N-1" single failure situation is considered. Therefore to meet P2/6 requirements with the current firm capacity at 35.4MVA and an additional transfer capacity of the site limited to 2MVA the allowable limit is 37.4MVA.

Figure 6.11[7] shows the single line diagram of the network with the conventional reinforcement option and the alternate approach of using ESS and Figure 6.12 [7] shows the base case where the firm and transfer capacity is exceeded.

The traditional reinforcement would require the installation of a third line with a 38MVA transformer to provide an additional 35.4MVA of firm capacity which is significantly above requirements. It is assumed that with battery energy storage systems (BESS) an improved power factor can be achieved via power electronics and thus the MVA requirements can be translated to MW and MWh requirements for BESS.

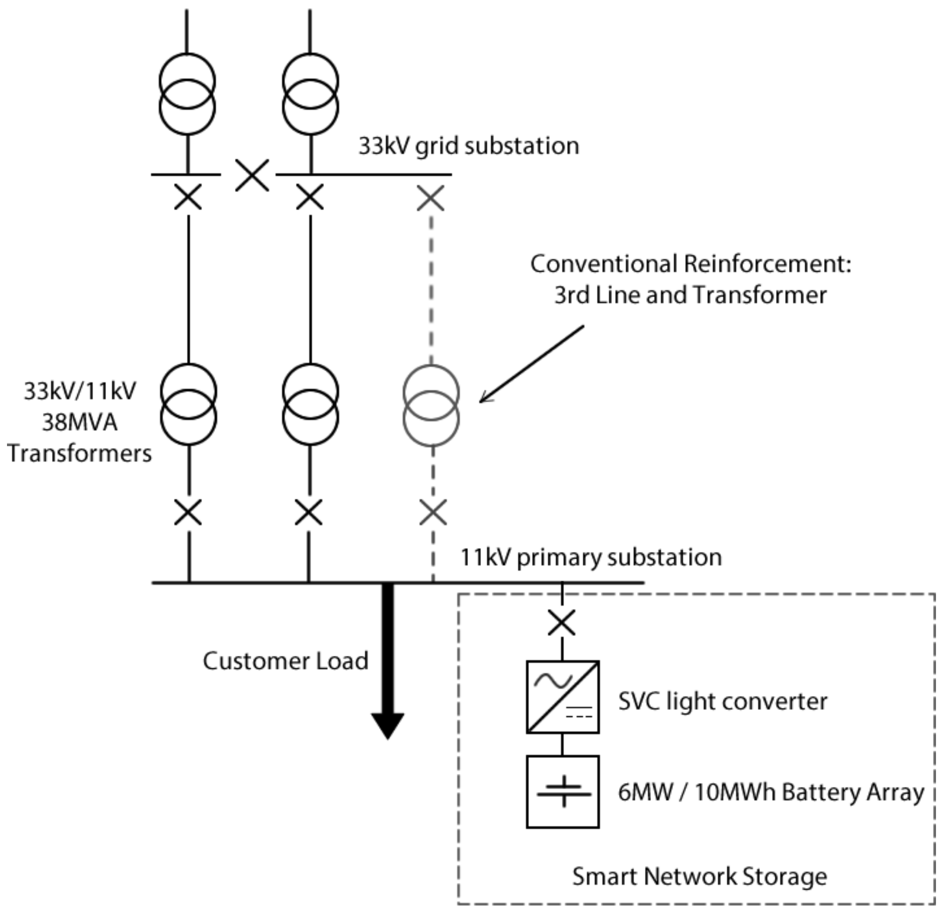

33kV grid substation
Conventional Reinforcement: 3rd Line and Transformer
33kV/11kV 38MVA Transformers
11kV primary substation
Customer Load
SVC light converter
6MW / 10MWh Battery Array
Smart Network Storage

Figure 6.11. Single line diagram of Leighton Buzzard distribution network substation

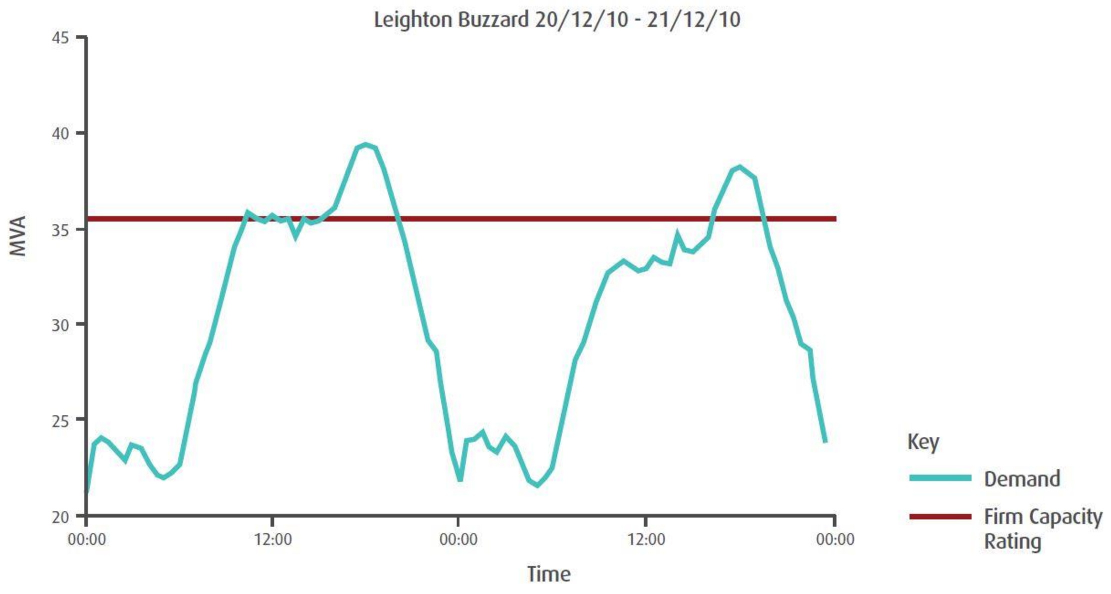


Figure 6.12. Leighton Buzzard capacity violations on two occasions

UKPN estimated that for the highest peak demand of about 40MW the BESS required is a 6MW/10MWh system. For control UKPN indicated the possibility of implementing a closed loop algorithmic control similar to a heating thermostat allowing the system to respond and adjust to meet a set-point [7], which describes a fixed set-point control (SPC) scheme.

This case study implements the set-point scheme as described to investigate the impact on the demand profile and also investigates the impact of using the evolutionary genetic algorithm - combinatorial optimisation (GACO) formulated with different block configurations. The set-point results are also evaluated using the fitness function for scoring the GA results and then the results of both schemes are analysed and compared.

6.4.2 Test data and source

The data used for the test were obtained from the design and planning considerations document from the UKPN [7]. Winter is the season with the most likelihood of a constraint violation and the data provided is for a winter day where the network capacity of 35.4MW has been exceeded with a peak demand of 39.66MW at 1900hrs and for eight other time periods. The hourly demand data used as input is given in Table A.5 which also shows the values converted to per unit on a base of 39.66MW.

Table 6.2 shows the evolutionary GACO parameters used for the case study including the system description, ESS parameters and set-point control parameters.

The GACO was tested with different maximum energy block number constraints – 7 blocks (10MWh split up into 7 different units) and 10 blocks (10MWh split into blocks of about 1MWh each). The variation was set to observe the impact of different energy block numbers on the result evolution.

The implementation of the set-point scheme attempts to charge the battery at its rated power when it’s fully discharged and according to the remaining state of charge as the capacity fills up. This is the closed loop method that operates like a thermostat.

Table 6.2. Case Study Network and Computation Parameters

Parameter	**Value**
Firm Capacity	35.4MW
Available transfer capacity	2MW
Peak demand value exceeding firm capacity	39.6MW
BESS rated capacity	10MWh
BESS rated power	6MW
BESS roundtrip efficiency (assumed)	80%
Set-point control switch value	35.4MW i.e. network firm capacity
GACO maximum demand	39.6MW i.e. peak demand
GACO ESS unit number constraints	10 units of 0.1kWh (GACO10) and 7 (GACO7)units of various capacity
GACO charging and discharging method	Worst Fit bin packing ascending ordered items
GA population size	20
GA maximum number of generations	15000
Mutation rate	50% (with constraint to prevent weaker chromosomes from returning to population)

6.4.3 Results and Discussion

The summary of results of the set-point control and the final evolved schedules with different ESS unit numbers from the GACO algorithm is shown in Table 6.3 and in Figure 6.13. The results show that in all cases the GACO performs better overall than the closed-loop set-point control method – before and after the further GA optimisation.

The SPC has a fitness score of 0.55 while the GACO10 has an initial and evolved fitness score of 3.13, and 3.22 respectively. The GACO7 has initial and final evolved score of 2.85 and 3.15. The demand evolution is shown graphically in Figure 6.14 and Figure 6.15 as the shape changes over several generations. Figure 6.16 is a scatter plot showing how the peak demand reduces over several generations, and Figure 6.17 shows gradual the reduction of the demand margin.

Table 6.3. Results Summary and Comparison

	Base Case	SPC	GACO7	GACO10
Peak demand (MW)	39.66	39.08	36.59	36.51
Trough demand (MW)	21.30	22.20	23.51	23.82
Demand margin (MW)	18.36	16.88 (-8.1%)	13.08 (-28.8%)	12.70 (-30.8%)
Peak demand reduction (MW)	-	0.58 (1.5%)	3.07 (7.7%)	3.15 (7.9%)
Score	-	0.55	3.15	3.22
Initial ESS SOC	-	0%	0%	0%
Final ESS SOC		100%	0%	0%
Energy consumed (MWh)	740.92	748.92	740.94	740.93

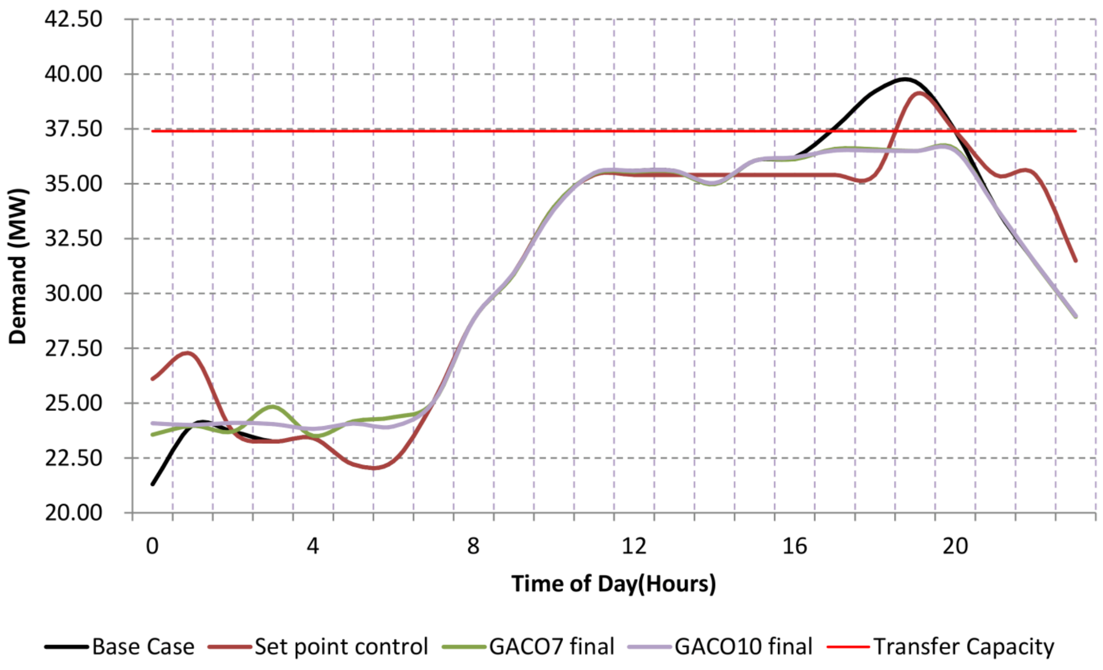


Figure 6.13. Base case and results of Set-point, GACO7 and GACO10 algorithms

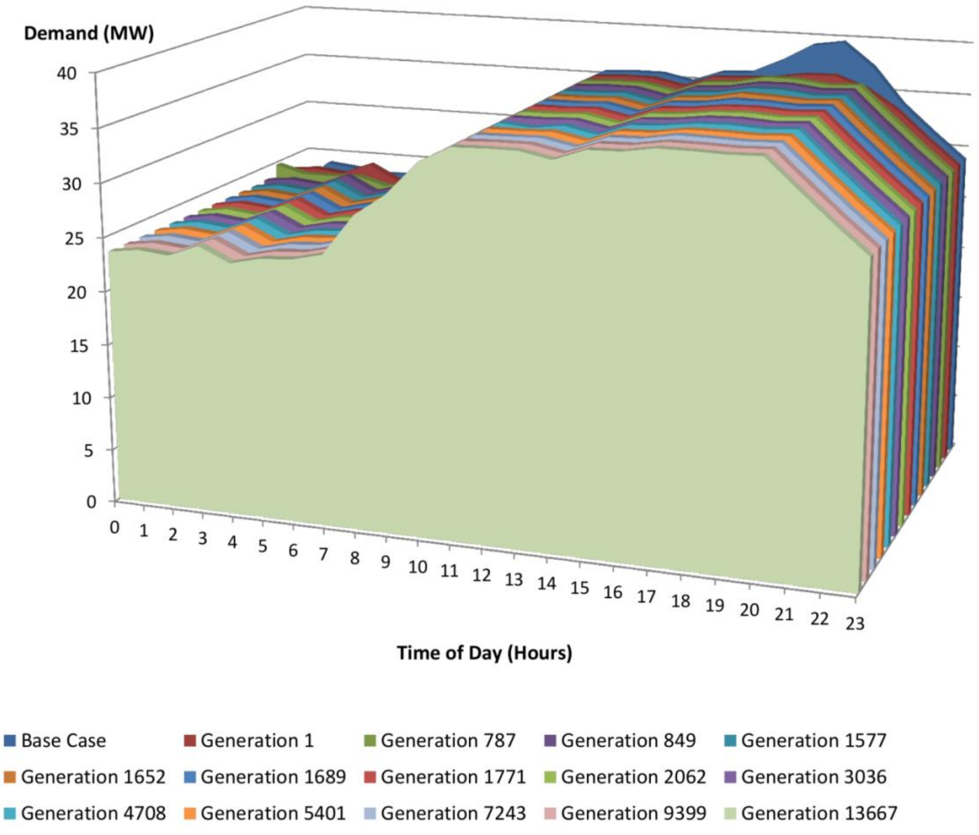


Figure 6.14. Demand shape evolution after application of GACO algorithm with 7 ESS units (GACO7) and 15000 generations

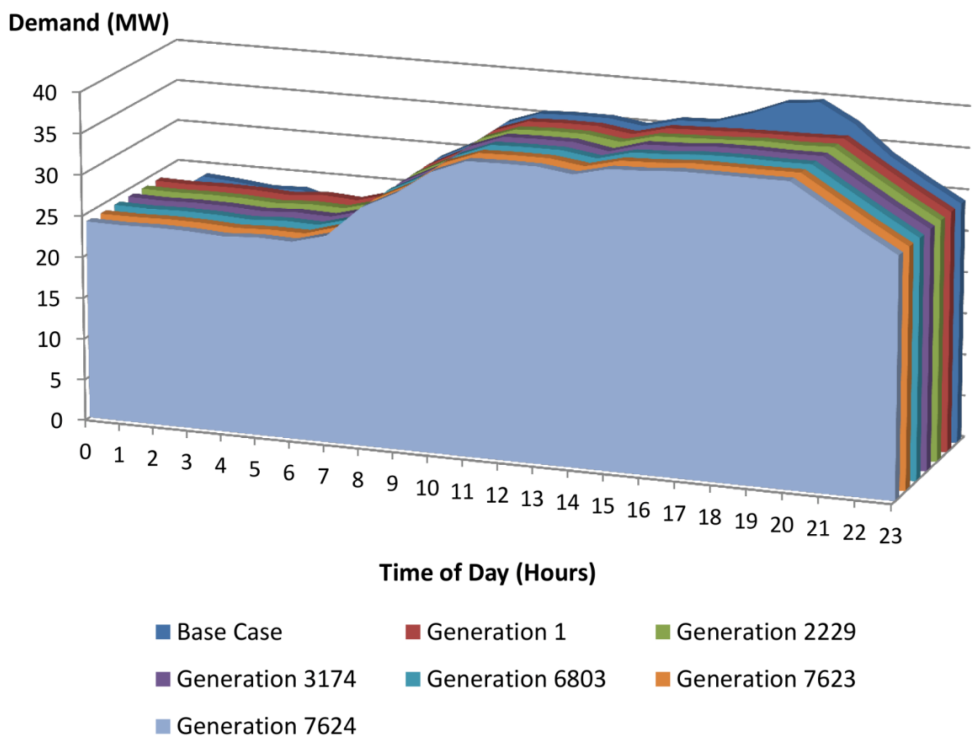


Figure 6.15. Demand shape evolution after application of GACO algorithm with 10 ESS units (GACO10) and 15000 generations

The SPC also demands more power (Figure 6.18) and consumes more energy at the end of the day because the battery SOC returns to 100% indicating inefficient operation, while the GACO methods discharge all the energy stored thereby resulting in proper demand shifting (Figure 6.19).

The set-point control method results in the least peak shaving of 0.58MW, and demand margin of 18.36MW using the ESS specified but the capacity is inadequate. In the first 2 hours of the day the ESS is charged to full capacity and does not have any effect on the profile for the next 9 hours while the demand is below the firm capacity of 35.4MW and therefore it does not discharge. From the 12th hour however the demand exceeds the firm capacity by 0.04MW, it discharges for the next 2hours and charges for an hour when the demand drops below the set-point. By the 18th hour the state of charge (SOC) drops to 7% after the previous discharge hours and by the time the main peak of 39.66MW is reached it can only be shaved by 0.58MW. This highlights a clear issue of set-point control not targeting peaks globally and not considering future events; the capacity of the ESS is used up when local peaks occur before the global peak.

The GACO shows that evolutionary algorithm does improve the quality of a schedule significantly. In both cases the final schedules after GA have been improved. The GACO10 peak shave evolves from 3.05MW to 3.15MW while for the GACO7 the values for initial and final are 2.78MW and 3.07MW respectively. The load levelling also improves as can be seen from the demand shape evolution (Figure 6.14 and Figure 6.15). For the GACO10 the demand margin starts at 12.81MW and ends at 12.70MW, and for GACO7 it starts at 13.48MW and ends at 13.08MW (Figure 6.17).

The 10-unit ESS produces a population whose members are all exactly the same because the packing algorithm will always produce the same result with the same items. The quality of the initial solution will tend to be better than solutions with fewer units as the granularity provides more opportunities for the packing operation to derive a flatter profile. The trade-off is that the GA can only produce a different chromosome via mutation and until the mutation produces a fitter chromosome it cannot benefit from improvement via crossover.

GACO10 also has more charging and discharging periods than the GACO7 and SPC schemes (Figure 6.18). This active involvement of the ESS results in the improved smoothening of demand as more targeting of peaks is done globally in the demand profile.

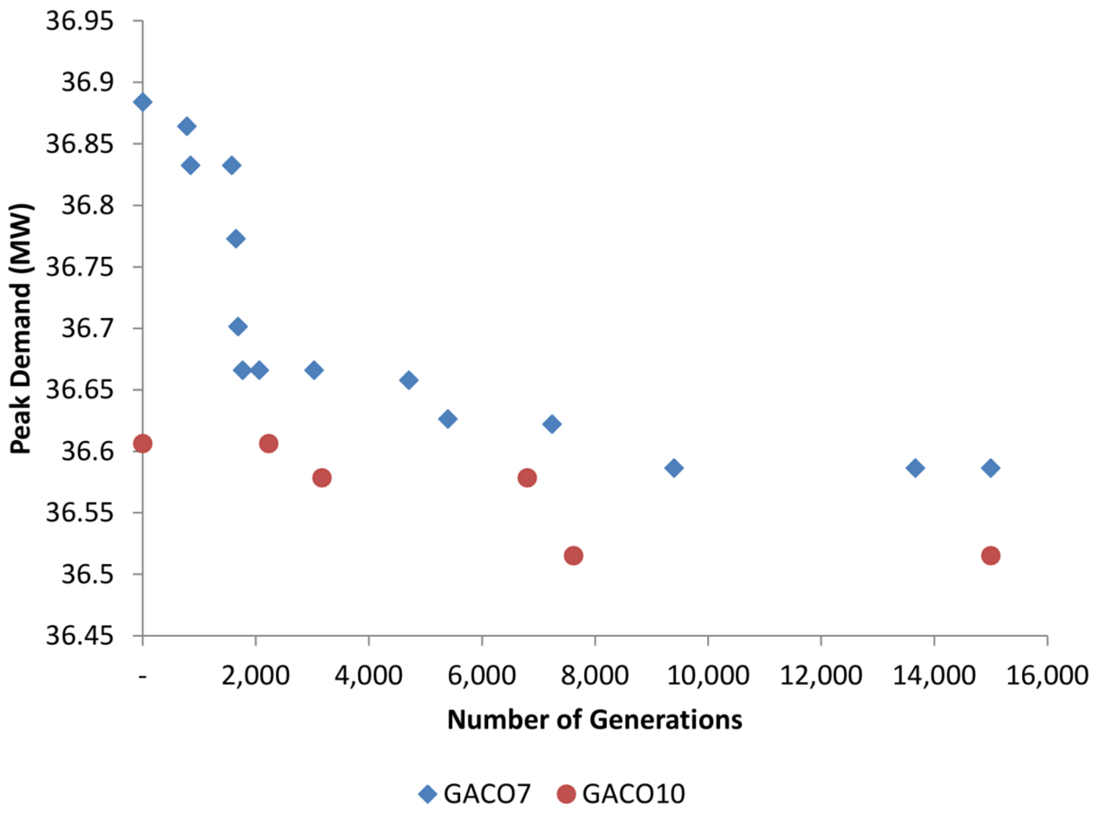


Figure 6.16. Evolution of peak demand after GACO algorithm

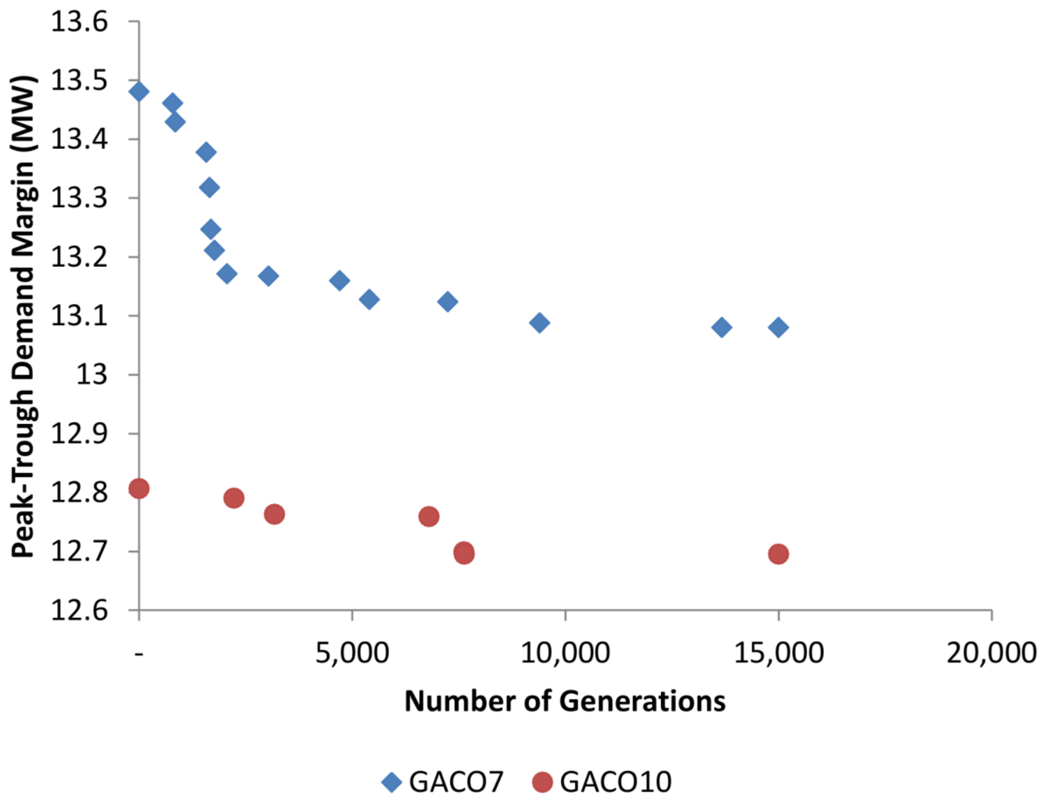


Figure 6.17. Evolution of demand margin showing improvement in load-levelling after application of GACO

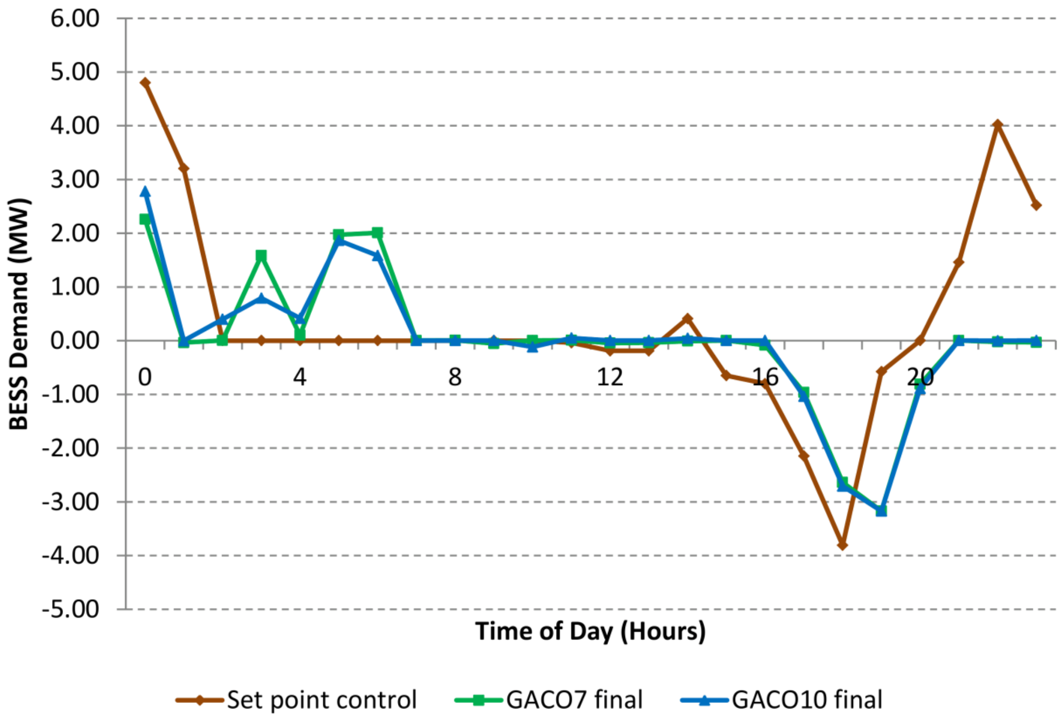


Figure 6.18. ESS power demand for set-point control, GACO7 and GACO10

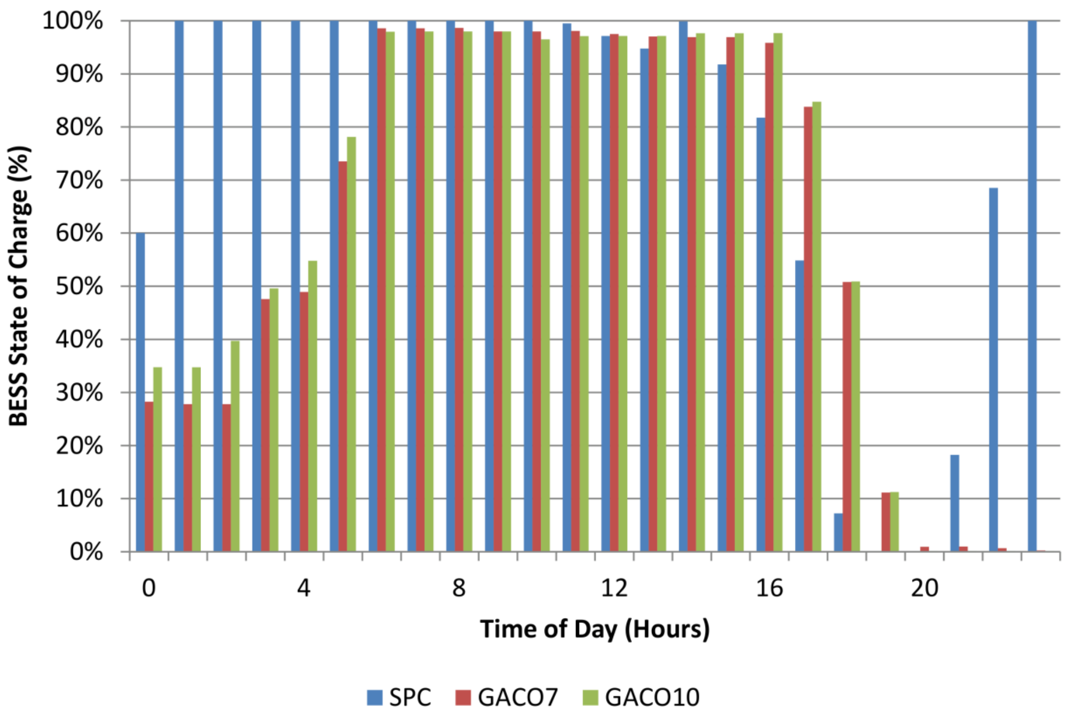


Figure 6.19. ESS State of Charge for set-point control, and final GACO7 and GACO10

GACO7 starts with a more diverse population but it was observed that it generally starts with a top score of 2.85 which is lower than the GACO10 initial top score of 3.13. The diverse population however allows the population to start crossover early on and combined with mutation the evolution of the population occurs more frequently. After a certain point the top scoring chromosomes become exactly the same and it also depends on only mutation to produce different solutions and more crossover opportunities.

In practice for the Leighton Buzzard SNS project the GACO schemes have shown better results than the SPC. In the case of the SPC the peak demand is not resolved with the 6MW/10MWh BESS and it will require a higher capacity to resolve the firm capacity violation, even with the 2MW transfer capacity activated. The GACO schemes have both brought down the peak demand to within firm capacity limits with the same battery specification and without using the transfer capacity, thereby leaving a transfer capacity headroom of 2MW at all times.

The GACO depends on accurate forecasts and at the distribution substation level the demand variation is not as much as at household level [38] hence this is feasible.

Given the upward trend of the GACO schemes schedule quality, the GACO algorithm can be allowed to run continuously in practice with different configurations and if it evolves an improvement in the schedule it may be implemented.

6.5 Case study and comparative analysis: Household demand scheduling with second-use EV batteries and tariff-driven demand response

6.5.1 Background

A case study is presented in which the representative residential electricity consumption data in southern Ontario, Canada is investigated for the effect of an ESS being installed and dispatched using the proposed algorithm. The case study features the base case and a number of scenarios where an ESS is dispatched using the combinatorial optimisation algorithm (CO).

The model setup is based on the detailed economic analysis by Heymans et. al in [51] where second use electric vehicle battery packs are used to provide load-shifting and demand response capabilities in a household with a view to reduce the cost of electricity and level the demand profile. The algorithm in [51] is a demand response (DR) scheme based on variable Time-of-Use pricing and using the ESS to shift energy to periods with lower energy prices to reduce the total daily cost of electricity.

A comparative analysis is carried out to compare the impact of the proposed combinatorial optimisation algorithm (CO) against the results of DR scheme in [51]. The DR method has a primary objective of reducing energy costs and load-levelling while the CO algorithm has the objective of peak reduction and load-levelling. The purpose of comparison is to investigate if the CO algorithm may be used to meet the cost reduction objective based on Time-of-Use pricing as suggested in the introduction. The impact of both algorithms on other characteristics of the household demand is also investigated.

6.5.2 Test data and source

The data used for the original research paper and this case study was obtained from the International Energy Association (IEA) Energy in Buildings and Communities program and is based on a three-year average [51]. The demand profiles used are the representative average household summer and winter daily demand profiles in a Canada household. The profiles are based on hourly intervals therefore 24 individual data points are used as shown in Table A.6. The ESS being used in the setup is a repurposed Chevrolet Volt EV battery which may no longer be used in an EV due to loss in capacity as a result of aging. The rated pack energy is

16.5kWh and the available energy based on warranties and industry targets for longevity is assumed to be 80 per cent of its rated capacity after 8 years of service in a vehicle [51].

The charge/discharge cycle efficiency is also affected by the aging and reduces the amount of energy available to be used in the battery pack. Based on the data sheet from the manufacturer’s website [104], the peak demand and power of the battery pack is calculated to be 4kW. This is calculated from the specified full charge time of 4hours at 240V to store 16.5kW h of energy. Table 6.5 shows the assumed capacity specifications of the ESS used in the study. In the case study setup, the battery is assumed to be fully discharged at the beginning of the cycle.

The CO algorithm requires all power and energy values to be scaled to a maximum of 10 per unit (PU), using a base of the maximum demand. In this case the maximum household demand is 1.74kW in both seasons however the maximum ESS demand is 4kW. Therefore all units were converted to PU values on a base of 4kW and then scaled to a maximum of 10 pu. On this scale, the peak battery power is 10 pu and the available capacity after a full charge and discharge cycle is 21.25 pu (originally 8.44kWh). The peak base case demand in both seasons becomes 4.35 pu. After the computation the results were scaled to the actual values.

The key parameters forming the basis of comparison in the base case versus the CO algorithm and DR versus CO are shown in Table 6.4.

Table 6.4. Parameters for comparison in case study

Case study	**Parameters compared – Basis for comparison**
Base case with no ESS versus CO	• Daily peak demand • Load levelling/ “flattening” of demand • Daily cost of energy
DR with ESS vs. CO	• Final peak demand and energy consumption • Load levelling • Savings on cost of electricity • Input parameters required for algorithm • Amount of charging and discharging required

Table 6.5. Battery Specifications for case study

Battery Specification	Value
Rated capacity	16.5 kWh
Available capacity	80% - 13.2kW h
Estimated peak power	4 kW
Charge efficiency	80%
Discharge efficiency	80%
Overall cycle efficiency	64%

For the purpose of the comparative analysis with the DR the same energy prices in [51] are used. The prices are derived from Waterloo North Hydro tariffs which implements three price periods – low ($0.067 per kWh), medium ($0.104 per kWh) and high ($0.124 per kWh). All prices given are in Canadian dollars. The daily tariff periods for summer and winter are shown in Figure 6.20.

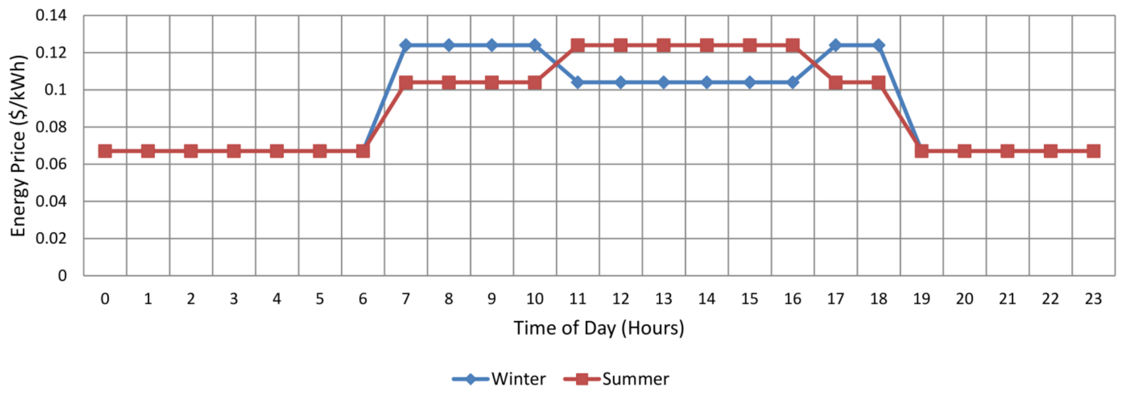


Figure 6.20. Time of Use Tariffs for Case study and comparative analysis

The algorithm was tested in three different scenarios for both summer and winter. Each scenario had a different number of energy blocks for the ESS to be packed into the demand "bins". A different viable solution with different characteristics may be obtained each time the algorithm is run as a result of the heuristic computation of the subset sum for the ESS capacity.

Using integer values only for the energy block subset sum the maximum number of blocks available is 21. For the summer computations 15, 18 and 21 energy block combinations were tested and for the winter computations 16, 19 and 21 block combinations were tested.

6.5.3 Results and discussion

Combinatorial Optimisation method compared to Base Case

The summary of the results are presented in Table 6.6, Table 6.7 and Figure 6.21 - Figure 6.22. For the case study against the base case, it can be seen that peak demand is reduced in some of the cases with the best shaving being 0.37kW in the 21 block case for both seasons and the highest increase being 0.38kW in the winter 16 block scenario.

The energy stored during the charge cycle is not always used up during the discharge cycle and that leads to the increase in demand and energy consumption. It can be seen that as the number of blocks increases the amount of energy used in the discharge cycle also increases thereby improving the utilisation of the ESS. The best results in all parameters are obtained in the 21 block scenario. Although the amount of energy consumed by the household daily increases by 1.67% in both cases the peak demand reduces by 22.29% (summer) and 21.26% (winter) as the ESS is used to redistribute utilisation and move consumption from the highest time periods to the lower time periods.

The daily cost of energy also reduces by 5.14% (winter) and 0.14% (summer) as a result of the ToU pricing and the time-shifting of demand to periods of lower consumption and lower tariffs. Since the tariffs are set based on consumption, then load-levelling also flattens the tariff indirectly by moving demand to periods with lower energy prices. The results also show that the difference between the peak demand and trough demand reduces by 26.67% (summer) and 35.56% (winter) after the algorithm has been used, which indicates a significant amount of levelling in the demand profile.

A general trend of improvement in peak reduction and cost saving can be observed as the number of blocks increases. This is expected because the bin-packing algorithm is provided more opportunities to shift demand around the daily demand profile with smaller sized blocks in the levelling operation. The peak increase in some cases is also due to the placement of a larger block of energy in a time period which cannot otherwise be split up and distributed into other time periods to level the demand.

The results show that a reduction in peak demand, a flatter demand profile and savings in energy cost may be attained in the base case when an ESS is dispatched using the Combinatorial Optimisation algorithm.

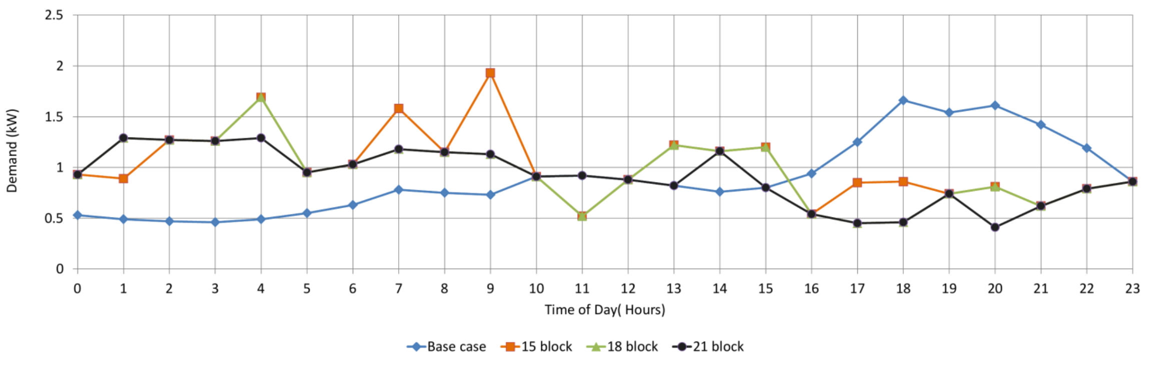


Figure 6.21. Summer Demand using Algorithm to dispatch ESS in different energy block configurations

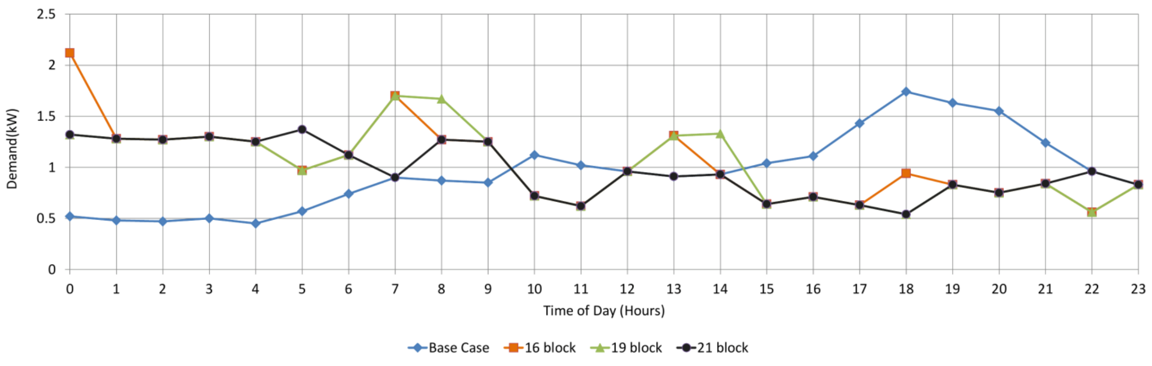


Figure 6.22. Winter demand using algorithm to dispatch ESS in different energy block configurations

Demand response results compared to Combinatorial Optimisation algorithm results

The DR algorithm is used to benchmark the performance of the CO algorithm with an energy cost saving objective as that is the primary objective of the DR scheme. Table 6.6 and Table 6.7show that both methods result in a saving in the daily cost of energy; the DR resulted in 6.87% and 10.69% of savings (summer/winter) while the CO resulted in about 0.14% and 5.14% of savings (summer/winter) in energy cost in both seasons against the base case cost of energy. The difference is approximately 5% less savings in the CO method compared to the DR method.

The better performance of the DR algorithm is expected since that is its main objective; however the savings in daily energy cost comes at the expense of other demand characteristics. The peak demand is increased in the DR method by 35.54% (summer) and 41.95% (winter) while it is *reduced* in the CO method (26.67% and 35.56%); the CO method also performs better in demand levelling as the DR method cuts off energy consumption totally in the periods of highest energy pricing, the 11th – 15th hour in summer (Figure 6.23) and the 7th – 10th hours in winter(Figure 6.24), and increases it in the cheaper periods creating new peaks and more variation in demand.

This comparison highlights an issue with ToU-based demand response at a household level with a cost saving objective and without significant constraints on demand. The cost saving benefits gained by the consumer taking advantage of the ToU tariff may lead to new problems for the distribution network operator (DNO). For instance if the same DR scheme is used for 200 homes connected to the same feeder then the congestion period will move to the lower price periods leading to the introduction of a new ToU tariff to decongest the network. The same DR scheme applied to the new ToU tariff will lead to a new congestion period and so on. The accumulated higher peak demand will also require a reinforcement of the network assets to handle the new peak.

The CO scheme with demand shaving and load levelling will therefore be more beneficial to both the consumer and the distribution system operator as there will be savings for the consumer and an improvement in the demand characteristics for the DNO.

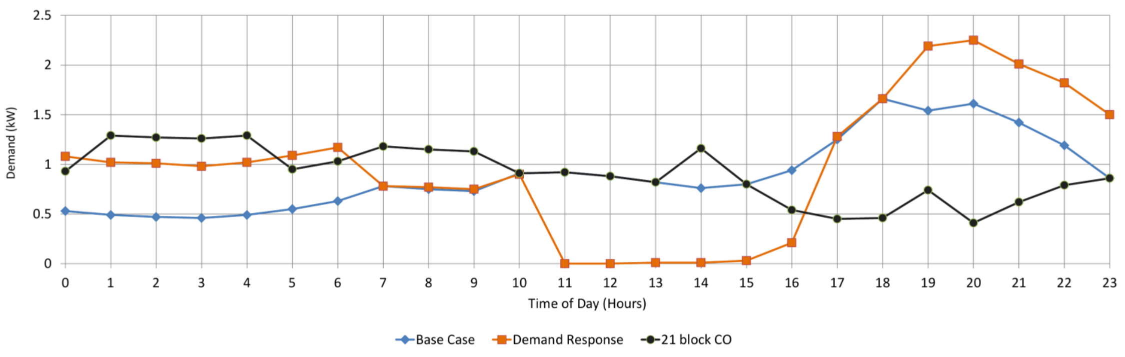


Figure 6.23. Comparison of Demand Response scheme, 21 Block CO and Base case in Summer

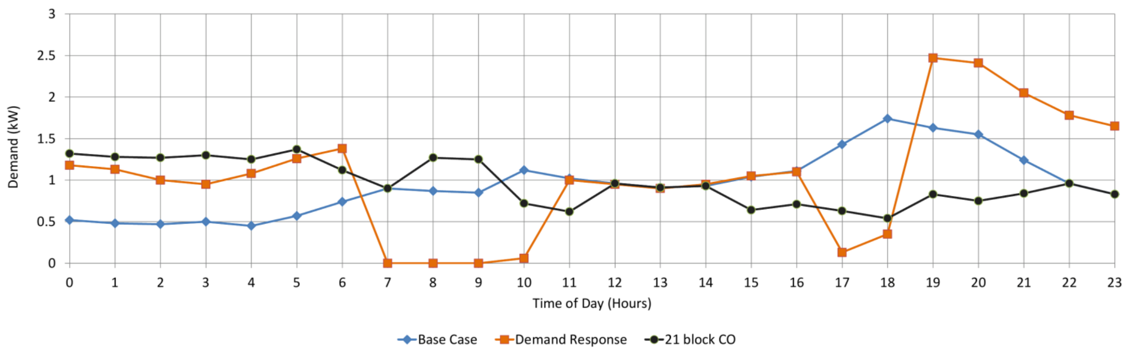


Figure 6.24. Comparison of Demand Response scheme, 21 Block CO and Base case in Winter

Table 6.6.Summer characteristics for case study and comparative analysis

	Base case	**15 block**	**18 block**	**21 block**	**Demand Response**
Peak – Trough margin (kW)	1.2	1.41	1.24	0.88	2.25
Peak demand (kW)	1.66	1.93	1.69	1.29	2.25
Peak Difference(kW)	-	+0.27	+0.03	-0.37	+0.59
Total consumption (kWh)	21.44	24.64	23.04	21.84	23.54
Daily energy cost ($)	1.95328	2.23488	2.05368	1.95048	1.81918

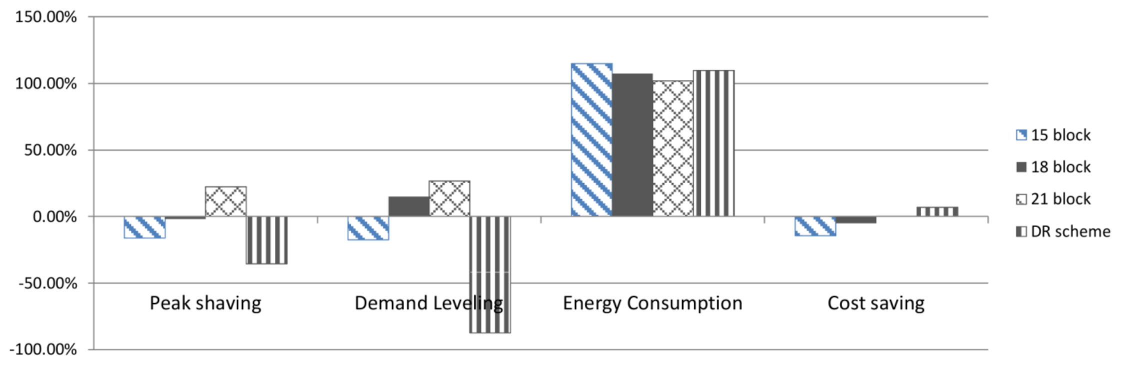


Figure 6.25. Relative summer characteristics for case study and comparative analysis

Table 6.7. Winter characteristics for base case and test scenarios

	Base case	16 block	19 block	21 block	Demand Response
Peak – Trough margin (kW)	1.29	1.56	1.16	0.83	2.47
Peak demand (kW)	1.74	2.12	1.7	1.37	2.47
Peak Difference from base case peak (kW)	-	+0.38	-0.04	-0.37	+0.73
Total consumption (kWh)	22.82	24.8	24.4	23.2	24.83
Daily energy cost ($)	2.1437	2.22396	2.21196	2.03356	1.91454

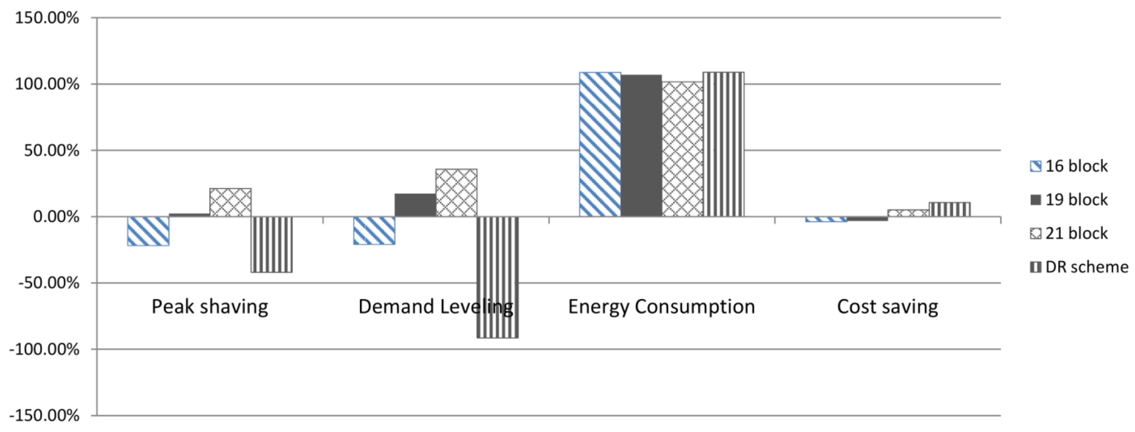


Figure 6.26. Relative summer characteristics for case study and comparative analysis

6.6 Summary

Four case studies and comparative analyses have been presented to test the operation of the evolutionary combinatorial optimisation algorithm developed. The case studies were:

- A study on Riverside Community, Stirling Scotland which showed that the peak demand and demand variation of a community of 1,015 households could be reduced using the algorithm.
- A comparative analysis with another verified algorithm based on dynamic set-point control was used to benchmark to performance and verify the accuracy of results obtained.

- An analysis of a network congestion management project was carried out by dispatching the specified ESS in the project using the evolutionary combinatorial optimisation algorithm and compared to results of using a closed loop set-point control scheme. The test showed how the evolution of schedules visibly improved results.
- A comparative analysis with a demand response scheme setup for energy cost savings in a household showed that although the demand response scheme resulted in higher savings the demand characteristics were greatly improved using the evolutionary combinatorial optimisation algorithm which also resulted in savings.

In all cases the algorithm performed at least as well as expected and in all cases there was a clear peak reduction and load levelling.

7 Web-based Power Systems Simulation using PHP: Tests, Results and Discussion

This chapter presents the results obtained from testing the web-based power systems analysis (WBPSA) library developed using PHP. The tests use standard networks from the Institute of Electrical and Electronics Engineers and networks obtained from a power systems analysis textbook [3] with verified results.

The test methodology is based on result comparison and the ability of the library to accurately produce on the same results on the same set of inputs using the same power flow solution. The library has a basic user interface for displaying results in a web browser and network input is made in extensible mark-up language (XML) in a text editor rather than in a graphical user interface (GUI).

The objectives of the tests are as follows:

i. To determine if the results from the library are consistent with verified results
ii. To determine the resource usage and performance of the library in terms of computation speed and memory usage
iii. To determine the impact of network size on the performance and any correlation between small changes or large changes in size and performance.

These parameters were chosen to be observed because of the most likely use-cases of this library which will involve using individual modules as part of a larger application residing on a server that receives multiple requests. In this kind of setup the cost of including the library will be determined on the basis of these parameters.

The PHP application for these particular results was run on a shared remote web server with the software specification shown in Table 7.1. There are several other configuration parameters however these are the relevant parameters to this implementation.

Table 7.1. PHP server software specification

	Item	Value
1.	PHP Version	5.2.17
2.	Web Server	Apache 2.0.64
3	Operating system	Linux

7.1 Power flow study using Newton-Raphson method: Tests and Results

7.1.1 Test data

The implementation of the methodology was tested by running a load flow analysis on different sized networks obtained from [3] and comparing with the results given. The computation of the load flow and network modelling in [3] was done using MATLAB programs.

The network sizes tested are 2-bus, 3-bus, 6-bus, 26-bus, and 30-bus networks. The networks were chosen as specified to test smaller networks in succession and larger networks in succession and the difference in performance over a large change in network size between smaller and larger network sizes.

7.1.2 Results and Discussion

The solutions for 2-Bus, 3-Bus, 6-Bus, 26-Bus and 30-Bus networks obtained from [3] and [67] were found to be consistent and accurate according to the results provided when the Newton-Raphson load flow was performed on them.

Because of the modularity of the application each of the functions (such as formation of admittance matrix, initialization of network, formation of Jacobian matrix, etc.) can be taken and used separately as part of a different application, thus the parameters were observed on a component/function basis.

The parameters observed also vary according to network size, network elements, convergence criteria and limits imposed. In this case however the number of buses in the network was used as an indicator of increasing complexity. Table 7.2 shows the average script execution time per iteration, and

Table 7.3 shows the amount of memory allocated to the different operations involved.

It is important to note that these script execution times and memory requirements are not related to the specifications of the end user's computer and are not affected by the performance of the end user's computer. This means the time and memory requirements for each process on the same server will be similar for all users. The sole requirement for using the PHP power system application is access to the application server via a web browser and the performance will depend fully on the server. The centralized system also means that updates to the power systems simulation program are immediately accessible by all users. For example, electrical engineers on the field accessing the server through their devices to perform some analysis will always have the latest version of the application and will have a similar experience of the process in terms of performance.

Table 7.2: Average Script Execution Time Per iteration (over 5 requests, in seconds)

	2-Bus network (4 Iterations)	3-Bus network (3 Iterations)	6-Bus network (4 Iterations)	26-Bus network* (3 Iterations)	30-Bus network* (4 Iterations)
Reading network data into application using XML	1.4529×10^{-3}	2.2199×10^{-3}	2.5980×10^{-3}	5.000×10^{-3}	5.0011×10^{-3}
Formation of admittance matrix	0.1640×10^{-3}	0.3569×10^{-3}	1.5819×10^{-3}	14.0011×10^{-3}	18.0020×10^{-3}
Formation of Jacobian matrix	0.2669×10^{-3}	0.3750×10^{-3}	2.1381×10^{-3}	21.0021×10^{-3}	27.0021×10^{-3}
Formation of power mismatch vector	0.089883×10^{-3}	0.1449×10^{-3}	0.5762×10^{-3}	3.9999×10^{-3}	5.0011×10^{-3}
Solution of sets of linear equations to obtain step solution (time per iteration)	0.6709×10^{-3}	1.1210×10^{-3}	12.8656×10^{-3}	570.0571×10^{-3}	930.0928×10^{-3}
Total including other functions	7.29513×10^{-3}	11.159×10^{-3}	67.9371×10^{-3}	1804.1799×10^{-3}	3845.3848×10^{-3}

* *Includes generator reactive power control*

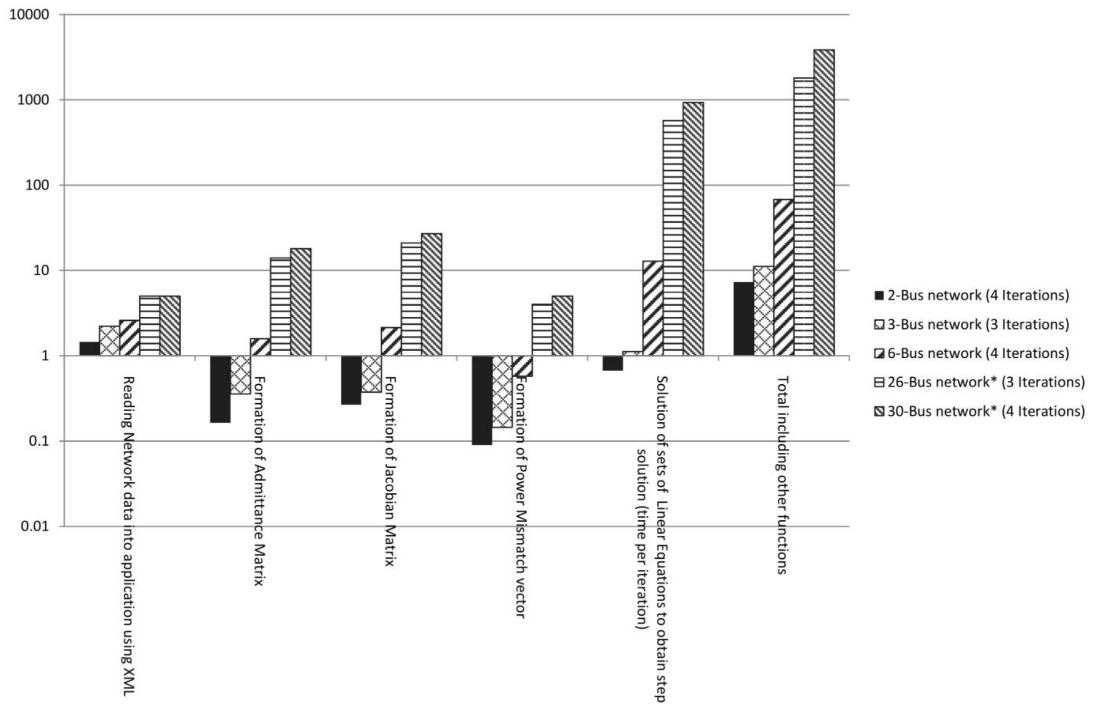


Figure 7.1. Timing of power systems simulation scripts in milliseconds on a logarithmic scale in base 10

Table 7.3: Server Memory allocation (in kB) – average for Jacobian matrix formation, power mismatch vector and step solution

	2-Bus network (4 Iterations)	3-Bus network (3 iterations)	6-Bus network (4 Iterations)	26-Bus network* (3 Iterations)	30-Bus network*
Reading Network data into application using XML	36.1171	41.3047	59.4180	207.6796	204.1953
Formation of admittance matrix	2.8555	4.5898	13.5898	245.8593	357.6406
Formation of Jacobian matrix	4.2148	5.9982	18.6299	472.2343	641.7265
Formation of power mismatch vector	1.2598	1.3164	1.9833	7.7578	8.8828
Solution of sets of Linear Equations to obtain step solution (per iteration)	1.1816	1.1497	1.42678	4.9531	5.4843
Total including other functions	226.3867	233.5703	266.3398	727.8984	854.7890

* *Includes generator reactive power control*

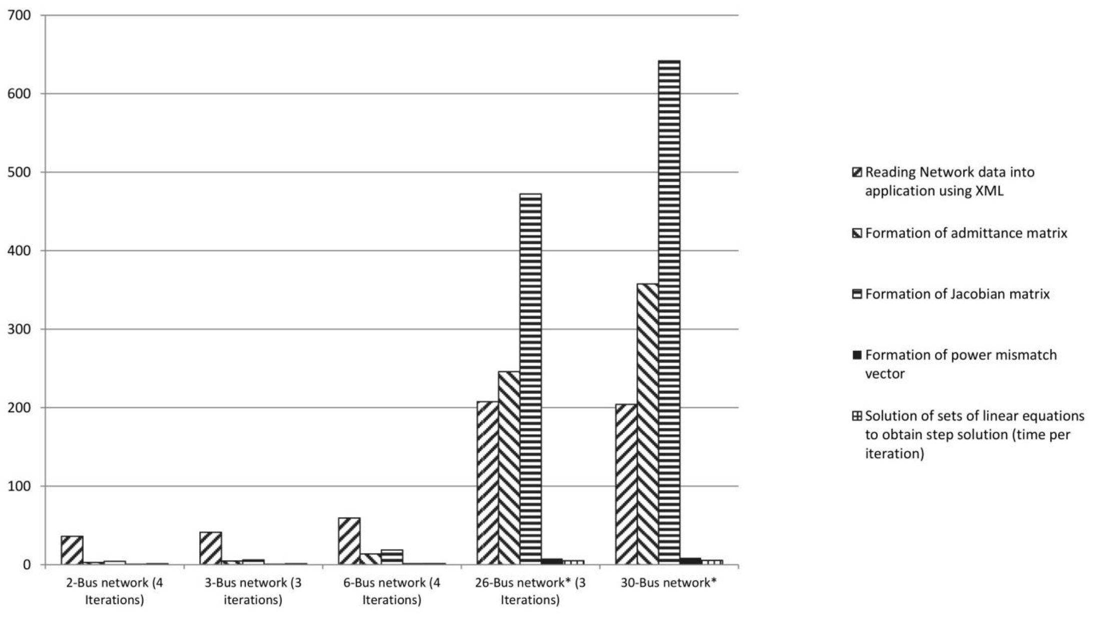


Figure 7.2. Server Memory Allocation for functions (in kB)

Reading network data into application using XML

It was observed from the tests that as the network size increased the time for reading network data into the PHP application did not increase significantly. The 2-bus network is contained in an XML file only 0.468kB in size while the 30-bus network has a file size of 7.26kB, and the average execution time for this operation was 1.4ms and 5.0ms respectively. This implies that the XML parser built into PHP is efficient and handles larger files without scaling the time for the operation proportionately.

Formation of admittance matrix

In forming the admittance matrix there is more of a linear increase in the time taken for the operation, as it goes from 0.164ms for the 2-bus up to 18.00ms for the 30-bus network. The increase between the 2-bus and 6-bus in terms of size is similar to the 26-bus to 30-bus networks however the change is about 1ms for the former group and 4ms for the larger group, implying an increasing trend as the size increases. The scale of memory consumption between the smaller and larger networks is also clearer in this case as the memory goes up from 2.86kB for the 2-bus network to 357.64kB for the 30-bus network.

Formation of Jacobian matrix

This is the other major matrix operation in the process and it follows the trend of an exponential type of increase between the smaller sized networks and larger networks in terms of execution time and memory consumption. Between the 2-bus and 6-bus networks the timing starts from 0.2669ms and goes to 2.138ms, however between the 26-bus and 30-bus networks the jump is more significant at 21.00ms and 27.00ms respectively. This is also the case for the memory consumption; the formation of the Jacobian matrix is the most memory intensive operation in the process as the 30-bus network consumes 641.73kB of memory, and it also represents the widest gap between metrics for the memory operations of smaller networks and the larger networks, with the 2-bus network consuming only 4.21kB. Hence almost 500kB of additional memory is required for the operation, compared to 300kB for the next memory intensive operation and 4kB for the least intense.

Formation of power mismatch vector

The power mismatch vector formation appears to follow the same trend as reading the network data between the smaller and larger networks in terms of memory consumption i.e. the margin is not so wide at 7kB. The timing of the scripts however goes up slightly quicker between sizes of networks when compared to the XML parsing operation – the XML operation timing margin between 26 and 30-bus networks is 0ms but in the case of the power mismatch vector it is approximately 1ms.

Solution of sets of linear equations to obtain step solution (per iteration)

The operation that takes the most time to complete is the solution to the sets of the linear equations in all the sizes of the networks. In the case of the 30-bus network it takes almost 1 second more per iteration, which is almost double the amount of time it takes for the 26-bus network. This difference adds up as the number of iterations required for the solution increases. The memory requirement per iteration is surprisingly lower than all other processes hence it is an efficient algorithm used in the Math_Matrix class for solving the set of linear equations in terms of memory.

Total time including other functions

The total time of all operations in finding a solution is most influenced by the solution to the sets of linear equations. This is further amplified in cases where more iteration is required. In the 30-bus network for instance out of the average total time of 3,845.39ms the operation solving the linear equations takes up 3,720.37ms. The memory usage for all operations however is not scaled as highly as would be expected between the lower sized networks and the higher ones. The 2-bus network (a 2 × 2 matrix) uses a total of 226.39kB as compared to 854.79kB on the 30-bus network (a 30 × 30 matrix) which is not a relatively large difference.

The execution times shown serve as an indication of how much latency may be expected by including these scripts in an application. The execution time is most relevant for real-time online applications which will require quick results computation.

Based on the results, the time required for the same functions increases significantly between the relatively smaller networks (2-bus, 3-bus and 6-bus) and the larger networks (26-bus, 30-bus), such that it has to be represented on a logarithmic scale. The key areas where this jump is most visible is in the matrix operations and the vector operations which are carried out using third party classes, so this is an area which must be optimised either by rewriting the classes to improve performance or by comparison with other classes for similar functions. The generator reactive power control on the larger networks also accounts for some of the additional processing time. The relationship between the timing and size of the network may be derived by carrying out further investigations with more network sizes between the extremes.

The memory usage is relevant in cases where an application is hosted on a cloud computing platform which includes billing according to memory usage. It is also noteworthy that the memory allocation is constant for every step even in separate requests. In applications where there are multiple requests made to the server for computation, both parameters will be preferred to be low.

From the results the smaller networks require the most memory for reading the network data into memory. As the size of the networks increase, the higher memory requirement comes from the matrix operations. The vector operations memory consumption remains relatively low and scales well between the network sizes.

The additional time and memory consumption from the matrix operations as the network size increases is not surprising as the matrix size does increase by up to a square with each successive dimension (2x2, 3x3, 4x4 …). Rewriting the matrix operations class with this performance improvement objective in mind or developing a PHP extension for specifically for matrix operations are some possible ways to improve the performance.

The computation speed may be improved in a number of ways without changing the structure of the code or the server machine specifications. Among the options available are changing the web server application because factors such as the transfer rate, average request time, requests handled per second and wait time for response, some of which affect the latency, vary for different web servers including Apache [105] and NginX [89] (pronounced "Engine Ex"). A virtual machine such as Hip-Hop Virtual Machine HHVM [28, 29] also improves PHP performance significantly and is increasingly being used on web application servers.

One of the advantages of an open source solution such as PHP is that a lot of third party applications exist to provide functionality or improvements that are not included in the PHP core.

Furthermore, PHP comes with shell execution functions [46, 47] which are the functions used to execute programs running on a separate simulation engine in a 3-tier framework. These functions can be triggered as a fall-back mechanism or to provide functions too complex to be derived in PHP, thereby taking advantage of its versatility.

7.2 Balanced Short Circuit Fault study

The results of the performance of the library in executing the 3-phase short circuit study are presented in this section. The networks tested are 3-bus, 6-bus, 14-bus and 30-bus networks with faults occurring at a single bus at a time. The metrics that were observed include the time and memory consumed in the main functions of the fault computation which are as follows:

i. Formation of impedance matrix by admittance matrix inversion
ii. Computation of faulted node fault current
iii. Computation of faulted node fault voltage
iv. Computation of fault voltages on other nodes
v. Computation of fault current in other nodes

The accuracy of the results were verified from the results in the data sources [3]. The performance results are shown in Table 7.4 – Table 7.5 and Figure 7.3 – Figure 7.4. The calculation method includes the initialisation functions and a power flow solution however these are not added to the performance metrics shown.

Table 7.4. Average short circuit script execution times (in seconds)

	3-Bus	**6-Bus**	**14-Bus**	**30-Bus**
Formation of Impedance Matrix	5.9042×10^{-3}	30.3502×10^{-3}	170.0273×10^{-3}	250.1122×10^{-3}
Faulted node fault current	0.1329×10^{-3}	0.1402×10^{-3}	0.1493×10^{-3}	0.1673×10^{-3}
Faulted node fault voltage	0.1835×10^{-3}	0.1902×10^{-3}	0.1955×10^{-3}	0.2023×10^{-3}
Network fault currents	0.4111×10^{-3}	0.7541×10^{-3}	2.3853×10^{-3}	5.1821×10^{-3}
Network fault voltages	0.4631×10^{-3}	0.8288×10^{-3}	2.900×10^{-3}	6.3901×10^{-3}
Total including other functions	7.2399×10^{-3}	33.0615×10^{-3}	204.1634×10^{-3}	262.4112×10^{-3}

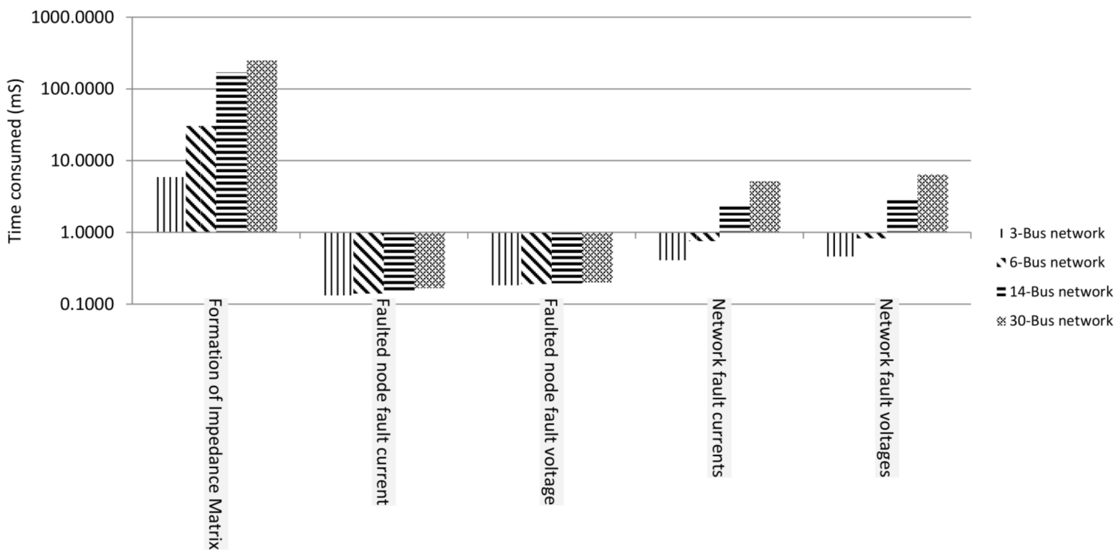


Figure 7.3. Average short circuit script execution times

Table 7.5. Average short circuit script memory consumption (in kilobytes)

	3-Bus	**6-Bus**	**14-Bus**	**30-Bus**
Formation of Impedance Matrix	60.2123	83.4221	353.6926	556.893
Faulted node fault current	1.2231	1.2023	1.4325	1.5732
Faulted node fault voltage	1.3229	1.4023	1.4772	1.5322

Network fault currents	3.5523	5.8566	23.7578	80.8828
Network fault voltages	3.8567	6.8526	35.6281	105.0039
Total including other functions	74.5673	102.7944	416.0482	746.6851

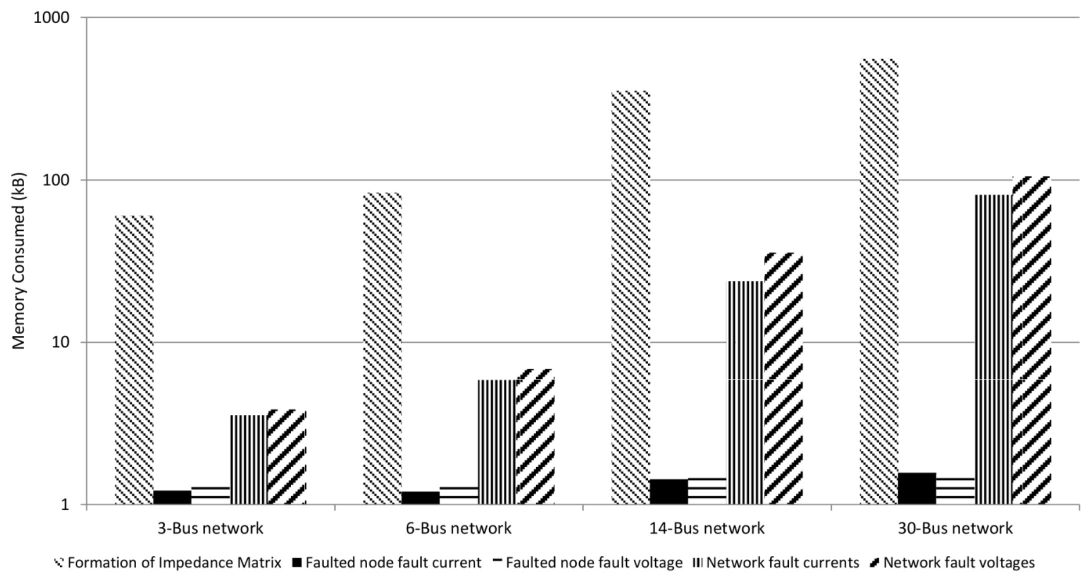


Figure 7.4. Average short circuit script memory consumption

Formation of impedance matrix

This is the process of inverting the admittance matrix which has been developed during the power flow solution. It is time and memory intensive as expected and as seen in the matrix processes of the power flow computation. As the size of the network increases so does the complexity of this process and therefore its impact on overall performance, however in this case it is not seen to be a squared increase between successive networks. This is likely as a result of the inversion algorithm built into the matrix class. This process takes up to 90% of the time and memory for the short circuit operations and any optimisation of resource usage must consider this function first.

Faulted node fault current

This process is a simple addition and division of values obtained from the impedance matrix and fault impedance and therefore it does not consume a lot of time or memory. It is also not related to the size of the network when a single bus is considered, therefore it can be seen that between the 3-bus network (0.1329ms, 1.2231kB) and the 30-bus (0.1673ms, 1.5732) there is only a small difference in the execution time and memory consumed.

Faulted node fault voltage

Similar to the faulted node fault current, this is a simple linear calculation of the voltage at the faulted node which is obtained using values of the calculated fault current and looking up the initial voltage result from the power flow solution. It can be seen that the execution time and memory range is similar to the fault current calculation for a single node and it also does not scale in square proportion like the network size. The difference between the 30-bus and 3-bus networks execution time is only ~0.02ms.

Network fault currents

This function calculates the fault current through all the nodes in the network using the linear calculations and it can be seen from the results that it follows a pattern for the number of nodes in the network and scales directly proportionally to the network size. This is because the process follows a simple loop and performs the same calculation for each node of the network.

Network fault voltages

Similar to the network fault current the resource usage for this process is directly proportional to the size of the network. This is also performed in a loop for each node in the network. Hence for an *n*-bus network the timing and memory are about *n* times the value for a single node calculation, which is expected.

7.2.1 Summary

The total time of all operations in finding a solution is most influenced by the solution to the sets of linear equations. This is further amplified in cases where more iterations are required. In the 30-bus network for instance out of the average total time of 3,845.39ms the operation solving the linear equations takes up 3,720.37ms. The memory usage for all operations however is not scaled as highly as would be expected between the lower sized networks and the higher ones. The 2-bus network (a 2 × 2 matrix) uses a total of 226.39kB as compared to 854.79kB on the 30-bus network (a 30 × 30 matrix) which is not a relatively large difference.

The execution times shown serve as an indication of how much latency may be expected by including these scripts in an application. The execution time is most relevant for real-time online applications which will require quick results computation.

Based on the results, the time required for the same functions increases significantly between the relatively smaller networks (2-bus, 3-bus and 6-bus) and the larger networks (26-bus, 30-bus), such that it has to be represented on a logarithmic scale. The key areas where this jump is most visible is in the matrix operations and the vector operations which are carried out using third party classes, so this is an area which must be optimised either by rewriting the classes to improve performance or by comparison with other classes for similar functions. The generator reactive power control on the larger networks also accounts for some of the additional processing time. The relationship between the timing and size of the network may be derived by carrying out further investigations with more network sizes between the extremes.

Due to the linear nature of the operations carried out in the short circuit study the bulk of the memory and time consumed is as a result of the impedance matrix formation. This was seen in the power flow solution as well and it accounts for about 80 – 90% of the total time consumption and ~80% of the memory consumption in the short circuit study.

The memory usage is relevant in cases where an application is hosted on a cloud computing platform which includes billing according to memory usage. It is also noteworthy that the memory allocation is constant for every step even in separate requests. In applications where there are multiple requests made to the server for computation, both parameters will be preferred to be low.

From the results the smaller networks require the most memory for reading the network data into memory. As the size of the networks increase, the higher memory requirement comes from the matrix operations. The vector operations memory consumption remains relatively low and scales well between the network sizes.

The additional time and memory consumption from the matrix operations as the network size increases is not surprising as the matrix size does increase by up to a square with each successive dimension (2x2, 3x3, 4x4 …). Rewriting the matrix operations class with this performance improvement objective in mind or developing a PHP extension for specifically for matrix operations are some possible ways to improve the performance.

The computation speed may be improved in a number of ways without changing the structure of the code or the server machine specifications. Among the options available are changing the web server application because factors such as the transfer rate, average request time, requests handled per second and wait time for response, some of which affect the latency, vary for different web servers including Apache [105] and NginX [89] (pronounced “Engine Ex”). A virtual machine such as Hip-Hop Virtual Machine HHVM [28, 29] also improves PHP performance significantly and is increasingly being used on web application servers.

One of the advantages of an open source solution such as PHP is that a lot of third party applications exist to provide functionality or improvements that are not included in the PHP core.

Furthermore, PHP comes with shell execution functions [46, 47] which are the functions used to execute programs running on a separate simulation engine in a 3-tier framework. These functions can be triggered as a fall-back mechanism or to provide functions too complex to be derived in PHP, thereby taking advantage of its versatility.

8 Conclusions and recommendations for future work

8.1 Energy storage system scheduling for peak shaving and load-levelling

8.1.1 Summary of findings

This study set out to address the issues faced by electricity distribution network operators (DNO) and consumers in managing problems caused by growing peak electricity demand using alternatives to network reinforcement which may be financially or logistically expensive. The problems that may arise include operational limit violations in existing equipment, loss of power, and inability to guarantee security of supply at all time periods.

Several methods are being used to operate a network such that peak demand is shifted to periods of lower demand allowing for more effective use of existing infrastructure, and also to reduce the margins between peak and trough demands thereby producing a levelled, predictable profile. These functions were defined as peak demand reduction (or peak shaving) and load levelling. The thesis showed with illustrations how these concepts differ and how one may sometimes produce the effect of the other. The thesis showed how energy storage systems (ESS) are currently in use for these functions using active network management (ANM) schemes which involves a set of practices that act pre-emptively to keep a network operating within limits. ANM usually relies on forecasts and the thesis showed the role forecasts can play in scheduling ESS for meeting set objectives in network operation. It also showed that ESS need to be scheduled to derive the full benefits as energy that is to be used must have been stored during a previous time period, also known as the inter-temporal nature of storage.

The different types of energy storage systems were reviewed and categorised according to their functions which include backup power supply, power quality improvement, demand time shifting, energy cost saving and mobility in electric vehicles. These functions can provide benefits at the electricity grid level or to a consumer. Some examples of scenarios were ESS are used on the grid were highlighted as well as scenarios for consumer use. The energy storage systems were classified according to the type of energy stored as mechanical, electrochemical, chemical, thermal and electrical. Another form of classification shown was according to the duration and frequency of use which impacts the purpose for which the ESS can be applied. The types of storage applicable for different objectives according to the discharge time (short, medium and long) and energy to power ratio and for the purpose of peak demand reduction and load levelling it was shown that battery energy storage systems are the most suitable at grid level and consumer level.

The existing ESS scheduling methods reviewed were set-point control, dynamic optimal power flow, dynamic programming, and demand response methods. The theory and operation principles of each of the methods were described along with the advantages and disadvantages of each set of methods. From the review it can be seen that the existing scheduling methods that exist have so far failed to address the impact of the amount of information controlled by and available to network operators and consumers. Most of the methods depend on generation cost or flexible time-of-use tariffs in addition to a demand forecast and ESS parameter specifications. In a deregulated market where generation and distribution of electricity are controlled by separate entities the DNO cannot control or sometimes access the generation prices to form a basis for optimisation, this is the same for demand response schemes where the DNO does not always control the flexible tariffs and cannot guarantee influencing consumer behaviour. Set-point control has advantages in versatility but is vulnerable to changes in future events and the best set-point may be difficult to determine.

In the formulation of the novel methodology to address these issues and perform optimisation based on only ESS parameters and a demand forecast the problem of ESS scheduling was defined as a form of a bin-packing problem and the manner in which the ESS energy is allocated was defined as a subset-problem. These are well-known combinatorial optimisation problems with heuristic solutions. The methodology based on the heuristic methods of solving these problems and extended them to suit the purpose of ESS scheduling particularly. The methodology modifies the subset-sum algorithm by adding a tolerance criterion to the backtracking tree-search method of solving it. Using the solution of the subset-sum problem as items to be packed the bin-packing problem was applied to a “binned” demand profile which undergoes specific transformations in charging and discharging phases. It was shown that different viable schedules may be produced using the methodology as a result of generating the energy allocations randomly from a bounded search space. The diversity of solutions allows for further optimisation that combines the best features of each schedule to produce an improved schedule. A genetic algorithm was developed for performing evolutionary optimisation of a “population” of schedules based on a fitness function that captures a peak shaving and load levelling objective.

To test the evolutionary combinatorial optimisation algorithm four case studies and comparative analyses were carried out. In the first case study a distribution network in Stirling, Scotland was evaluated and the peak demand was reduced by up to 38.5% and the demand margin between peak and trough demand was reduced by 50.2% using only the combinatorial optimisation algorithm without further evolutionary optimisation.

To verify the algorithm a comparative analysis was done with a verified demand tracking algorithm based on a dynamic set-point control scheme and it was seen that the combinatorial optimisation (CO) algorithm produced results in a similar range with the demand tracking algorithm with the CO algorithm showing slightly better results. In peak shaving for example the CO algorithm reduced the peak by 23% while the demand tracking method reduced it by 20%. The battery usage in the CO method is also less intensive as the charging and discharging periods were reduced by 50%.

The evolutionary algorithm was tested on a case study for a UK power networks smart network storage project based in Leighton Buzzard, UK. The genetic algorithm combinatorial optimisation was compared to a closed loop set-point control scheme. It was seen that results produced by the combinatorial optimisation schemes (7.7% and 7.9% reduction) were better than the set-point scheme (1.5% reduction in peak) and were further improved over several generations of evolution. In terms of demand levelling the set-point scheme improved the margin by 8.1% while the evolutionary combinatorial optimisation schemes improved it by up to 30.8%.The evolutionary algorithm tests also inspected the evolution of different numbers of energy blocks to see if there was any difference in rates of change and it was seen that although there was improvement of the schedules based on a lower number of blocks (7 blocks) the higher number of blocks (10 blocks) still produced better results after evolution but was more intensive on battery utilisation in charging and discharging. The conclusion was that evolution could indeed be used for improving the schedules.

A comparative analysis was performed with a demand response scheme designed to maximise savings in energy cost to investigate if the algorithm could produce good results in a time-of-use tariff system. It was found that the algorithm produced much better results in terms of demand characteristics and although it did not result in as much savings as the demand response scheme it still produced some cost savings as the demand flattening indirectly flattened the time-of-use tariffs.

8.1.2 Significance of findings

The following significant conclusions may be drawn from the findings:

i. The most significant conclusion is that peak demand reduction and load levelling optimisation may be performed effectively using only demand profile and ESS parameter information – the only information guaranteed to be available to network operators and consumers. The results were seen to be competitive in comparison with other methods without being computationally complex even though they are based on heuristic methods.

ii. These findings enhance our understanding of ESS scheduling by showing that it may be viewed in the form of the combinatorial optimisation problems as applied in this thesis. The implication is that any heuristic or exact solutions developed for the problems may be applied to ESS scheduling optimisation.

iii. These findings suggest a role for evolutionary algorithms to be used to further improve the results derived from any other exact or heuristic combinatorial optimisation solutions applied to the form of the problem defined in this thesis.

The research extends our knowledge of energy storage systems and will prove particularly valuable as configurable energy storage systems become more ubiquitous with the development of smart electricity grids and future electricity networks. The study suggests that more methods such as this will be developed as the problems associated with other methods become more apparent with the increasing penetration of ESS in electrical systems. Due to the difficulty of the problem and the non-linearity of demand the most likely solutions will be heuristic and may be based on combinations of other methods when the problem is reduced to a set of smaller problems.

As ESS becomes more prevalent in smart grid and residential use more services will be required to provide operational schedules for the ESS devices. This will possibly create a competitive market where methods such as the one developed in this thesis will be provided to end-users as premium remote services to improve the utilisation of their energy resources by third-party providers. A benefit of this possible scenario is that it may attract more research interest and investment in this area and expand the body of knowledge in the area.

8.1.3 Limitations of this study and recommendations for future work

A limitation of this study is that it focuses on off-line optimisation of ESS scheduling and can therefore not respond to significant changes in real-time events and major deviations to the forecast. This limitation exists in all other forecast-based methods where over-fitting is possible, with the exception of set-point control. Further research should focus on extending the method to include contingencies for unexpected changes in demand.

An issue that was also not addressed in this study is the optimisation of the number of energy blocks to be used from the subset-sum algorithm or the boundary of the search space. This will add a layer of complexity to the solution which may only be solved by repeated experimentation and regression analysis to extrapolate a trend relating the number of blocks to the quality of the solution. It would be interesting to assess methods for the optimisation of that preliminary stage in the optimisation process.

8.2 Web-based power systems analysis using PHP

8.2.1 Summary of findings

The methodology for developing a PHP library for web-based power systems analysis and simulation was presented in this thesis. Designing and planning networks for electricity distribution requires a significant amount of calculations and for decades computers have been used to carry out these calculations. Simulating the operation of the network allows engineers and researchers clearly understand the operating conditions of a network without physically assembling it and therefore simulation software packages are a vital part of an engineer's toolbox.

The thesis showed that traditional deployment of simulation software has been largely based on desktop personal computers or a local network of interconnected computers however with the growth of information and communications technology and ubiquitous internet access simulation software are also being deployed on the internet as web-based simulation packages.

It has been shown in this thesis that most web-based simulation packages for power systems analysis implement a 3-tier architecture where a programming language which is not compatible for the web but suitable for computation such as Matlab, C++ or Java is used as a simulation engine which carries out the actual computations in a back-end tier. In the front-end tier is the web browser such as Microsoft Internet explorer or Google chrome which is used for input and sending simulation requests and also viewing results. In the middle tier resides the web server running programs written in scripting languages such as PHP and ASP that passes information between the user on the web browser and the simulation engine in the back-end.

It was shown that this 3-tier methodology was used for a number of reasons including the inability of older versions of the web server programming languages to carry out the types of mathematical computations required for power systems analysis, and also because the majority of the legacy simulation packages written in general purpose programming languages incompatible with web servers already existed and could be re-used.

The review of existing software packages for web-based power systems analysis (WBPSA) including commercial, open-source, and research implementations described how systems based on Matlab and Visual basic programming are deployed in the 3-tier architecture and how commercial packages such as NEPLAN deploy their web versions also using the 3-tier architecture. The shortcomings of this approach include the multiple points of failure – failure could occur at the web server or the simulation engine or during data transfer processes; the licensing costs – most of the software used for simulation are premium packages which may be expensive; and additional server infrastructure requirements – a remote computer must be setup which is capable of running the simulation software.

The literature review section covered the fundamental principles of power systems studies to establish the mathematical and analytical requirements for performing power flow and short circuit fault studies. These requirements include Matrix operations, vector operations, complex number analysis, trigonometric functions and linear programming tools. These requirements are the same for every programming language that is to be used for power systems analysis and must be provided either natively or via third-party add-ons and extensions. The internal architecture of power systems simulation software was covered to describe the classes and functions that make up such a package including the data input functions, network initialisation and representation functions, analysis and solver functions, and output functions. In doing so one of the objectives of the thesis to establish the fundamental concepts required for general power systems analysis in any programming language was achieved.

A slimmer 2-tier methodology for WBPSA using PHP was proposed in this thesis, which is a novel approach to web-deployment of power systems simulation software. PHP remains the most popular web server programing language and recent versions of the language have some of the required capabilities for general power systems analysis outlined in the literature review. The advantages of this approach and of using PHP for WBPSA were covered and some of the advantages include platform flexibility, reduced licensing costs, extensive database support, suitability for smart networks and simplicity. The disadvantages of using PHP include the language flaws persistent from older versions, performance limitations and inadequate availability of libraries for power systems analysis.

The PHP WBPSA library was implemented using the core functions for mathematical analysis and third-party libraries for Matrix, vector and complex number analysis with modifications to suit the limitations of the libraries. A power flow solver based on the Newton-Raphson method was developed along with a balanced 3-phase short circuit fault solver.

The test carried out showed that the PHP library was accurate and efficient in solving power flow problems and for short circuit fault studies thus achieving the second objective for the 2-tier WBPSA framework. The tests used standard networks with verified results to investigate the accuracy of the results produced by the library and measure its performance in terms of computation time and memory consumption as the network size increased. The library passed the accuracy tests and the results produced were consistent.

It was found that the operations execution time was in the millisecond range and the memory consumption is in kilobytes for most operations. This suggests that the library is efficient and the server resource requirements are low. The most tasking operations in terms of time and memory were the matrix operations whose requirements increased significantly with small changes in network size such that they had to be represented on a logarithmic scale. This is expected as the number of elements in a matrix increases as a series of squares and not as an arithmetic sequence.

The library developed is a lightweight simulation tool that can be embedded in PHP websites to provide power systems analysis functions.

8.2.2 Significance of findings

This is the first time that PHP has been used in a 2-tier framework for web-based power systems analysis and simulation and the outcomes of this research has several practical applications outlined as follows:

i. The study makes a contribution to power systems simulation in general as PHP websites for researchers and analysts may easily embed the library and call its functions to provide interactive models of their work over the internet
ii. The research provides a framework for the exploration of web-based power systems simulation using other languages as newer languages are developed and grow in usage across the web
iii. The library may serve as a base for future web-based power systems analysis tools to build on and develop. The absence of licensing costs for PHP makes the library suitable as a starting point for building cost-effective alternatives to the commercial packages currently available

Using PHP to build this library has created a new option for programming power systems simulators and in doing so creates a thinner application with fewer resource requirements, fewer points of possible failure, high compatibility and suitability for web applications and versatility. This is not to say PHP is the best solution for web-based power systems analysis, but that it is a suitable and available option.

The use of a new programming language and method to deliver solutions which are already implemented in other programming languages and methods may not be associated with any major benefits when viewed in terms of the final outcomes for the same problems. However, the possible benefits to be derived reside in the smaller details involved in the process between the problem and the solution.

The new methodology provides another viable programming option for obtaining power systems solutions which researchers and engineers will find useful in their applications, and the minor advantages derived and slight changes in the process can scale up significantly to result in major benefits and perhaps a paradigm shift in how power systems simulation is approached.

8.2.3 Limitations of this study and recommendations for further work

The matrix operations were found to be the most demanding in terms of computation speed and memory requirements; these operations were carried out using a third party library. A limitation of this study is that the third party matrix library was not compared with other libraries to establish if any other libraries scale more efficiently with network size. It will be interesting to replace that module or modify it to observe the impact on the overall performance of the library.

There is also a limitation on the number of functions provided by this library and the lack of a graphical user interface (GUI) for a better user experience. It currently provides the most frequently used studies which form the basis of other studies and a basic input and output interface. Further development of the library should focus on including additional functions for other power systems studies and a user friendly web interface using JavaScript to provide network input and result output functionality.

Finally one source of weakness in this study which may affect its practical application is that the models using the functions provided by this library will have to be written in PHP to fully implement a 2-tier architecture. As many models for education and research are programmed using Matlab, it will be interesting to develop a Matlab to PHP interpreter that converts Matlab code to PHP which can conveniently use this library for web deployment of the models in a 2-tier framework.

Appendix A. Test data for evolutionary combinatorial optimisation algorithm

Table A.1. Riverside household count by occupancy type

Type of household	No. of households
Single adult	234
Single Pensioner Adult	159
Two adults	218
Two adults with children	156
Two pensioners	53
Two adults and at least 1 pensioner	92
Three adults	103
Total	1015

Table A.2. Winter daily electricity demand in Riverside community by household type in kW

Household Type	Single Adult	Single Pensioner	Two Adults	Two adults with Children	Two Pensioners	Two adults with a Pensioner(s)	Three Adults	Total (kW)
00:00 - 01:00	8.58	5.83	15.99	17.68	3.89	10.43	11.33	73.72
01:00 - 02:00	8.58	5.83	15.99	17.68	3.89	10.43	11.33	73.72
02:00 - 03:00	8.58	5.83	15.99	17.68	3.89	10.43	11.33	73.72
03:00 - 04:00	8.58	5.83	15.99	17.68	3.89	10.43	11.33	73.72
04:00 - 05:00	8.58	5.83	15.99	17.68	3.89	10.43	11.33	73.72
05:00 - 06:00	12.56	5.83	24.43	17.68	3.89	10.43	17.85	92.66
06:00 - 07:00	23.62	17.16	48.37	52.20	9.56	30.76	33.35	215.02
07:00 - 08:00	83.86	43.34	120.76	113.84	20.13	62.35	103.55	547.83
08:00 - 09:00	58.01	40.60	72.78	106.36	21.83	67.54	53.83	420.95
09:00 - 10:00	8.58	51.67	15.99	17.68	24.22	61.93	11.33	191.39
10:00 - 11:00	8.58	29.30	15.99	17.68	18.66	25.33	23.67	139.20
11:00 - 12:00	8.58	32.95	15.99	17.68	25.33	19.78	11.33	131.63
12:00 - 13:00	8.58	20.45	15.99	17.68	12.80	34.39	28.58	138.47
13:00 - 14:00	8.58	19.57	15.99	146.60	12.26	74.91	11.33	289.24
14:00 - 15:00	8.58	38.85	15.99	149.78	31.68	98.72	11.33	354.92
15:00 - 16:00	8.58	40.92	15.99	71.76	23.21	45.04	11.33	216.82
16:00 - 17:00	8.58	55.77	15.99	144.87	26.57	96.85	11.33	359.97
17:00 - 18:00	12.77	56.38	58.46	180.93	28.22	84.92	11.33	432.99
18:00 - 19:00	92.92	67.51	197.29	203.46	47.55	129.94	128.83	867.49
19:00 - 20:00	173.20	49.28	227.75	196.64	36.78	121.97	214.12	1019.74
20:00 - 21:00	127.49	61.47	187.42	186.91	41.57	95.40	156.74	857.01
21:00 - 22:00	140.48	55.66	218.42	145.83	45.18	77.79	169.49	852.85
22:00 - 23:00	113.84	29.68	175.24	145.40	11.84	66.47	128.26	670.73
23:00 - 24:00	85.78	29.68	130.08	139.31	11.84	66.39	94.37	557.46
Total	1036.07	775.22	1668.81	2160.68	472.57	1323.01	1288.60	8724.97

Table A.3. Riverside demand data after conversion to PU and scaling

Time of Day	Total Demand (kW)	Scaled demand (PU)
0000	73.72	0.722929
0100	73.72	0.722929
0200	73.72	0.722929
0300	73.72	0.722929
0400	73.72	0.722929
0500	92.66	0.908663
0600	215.02	2.108577
0700	547.83	5.372252
0800	420.95	4.128013
0900	191.39	1.876851
1000	139.2	1.365054
1100	131.63	1.290819
1200	138.47	1.357895
1300	289.24	2.836409
1400	354.92	3.480495
1500	216.82	2.126228
1600	359.97	3.530017
1700	432.99	4.246082
1800	867.49	8.506972
1900	1019.74	10
2000	857.01	8.404201
2100	852.85	8.363406
2200	670.73	6.577461
2300	557.46	5.466688

Table A.4. Representative average demand in a household in Denmark in spring

Time of Day	Demand (W)	Demand PU	Scaled demand (PU)
0000	340.1	0.38	3.8
0100	268.5	0.3	3
0200	250.6	0.28	2.8
0300	223.75	0.25	2.5
0400	223.75	0.25	2.5
0500	268.5	0.3	3
0600	331.15	0.37	3.7
0700	402.75	0.45	4.5
0800	438.55	0.49	4.9
0900	447.5	0.5	5
1000	456.45	0.51	5.1
1100	465.4	0.52	5.2
1200	456.45	0.51	5.1

1300	447.5	0.5	5
1400	429.6	0.48	4.8
1500	456.45	0.51	5.1
1600	492.25	0.55	5.5
1700	733.9	0.82	8.2
1800	895	1	10
1900	733.9	0.82	8.2
2000	698.1	0.78	7.8
2100	653.35	0.73	7.3
2200	617.55	0.69	6.9
2300	492.25	0.55	5.5

Table A.5. Leighton-Buzzard hourly demand used for tests in original and scaled PU values

Time of Day (Hours)	Base (MW)	Scaled (PU)
0000	21.3	5.370651
0100	24	6.051437
0200	23.7	5.975794
0300	23.25	5.86233
0400	23.403	5.900908
0500	22.2	5.597579
0600	22.35	5.635401
0700	25.06	6.318709
0800	28.82	7.266768
0900	30.93	7.79879
1000	33.94	8.557741
1100	35.44	8.935956
1200	35.59	8.973777
1300	35.59	8.973777
1400	34.99	8.822491
1500	36.05	9.089763
1600	36.2	9.127584
1700	37.55	9.467978
1800	39.21	9.886536
1900	39.66	10
2000	37.4	9.430156
2100	33.94	8.557741
2200	31.38	7.912254
2300	28.97	7.304589

Table A.6. Household demand data for Canada household

	Winter		Summer	
Time of Day (Hours)	**Original (kW)**	**Scaled (PU)**	**Original (kW)**	**Scaled (PU)**
0000	0.52	1.3	0.53	1.325
0100	0.48	1.2	0.49	1.225
0200	0.47	1.175	0.47	1.175
0300	0.5	1.25	0.46	1.15
0400	0.45	1.125	0.49	1.225
0500	0.57	1.425	0.55	1.375
0600	0.74	1.85	0.63	1.575
0700	0.9	2.25	0.78	1.95
0800	0.87	2.175	0.75	1.875
0900	0.85	2.125	0.73	1.825
1000	1.12	2.8	0.91	2.275
1100	1.02	2.55	0.92	2.3
1200	0.96	2.4	0.88	2.2
1300	0.91	2.275	0.82	2.05
1400	0.93	2.325	0.76	1.9
1500	1.04	2.6	0.8	2
1600	1.11	2.775	0.94	2.35
1700	1.43	3.575	1.25	3.125
1800	1.74	4.35	1.66	4.15
1900	1.63	4.075	1.54	3.85
2000	1.55	3.875	1.61	4.025
2100	1.24	3.1	1.42	3.55
2200	0.96	2.4	1.19	2.975
2300	0.83	2.075	0.86	2.15

References

[1] M. Wilks, Pöyry, and University of Bath, “Demand side response : Conflict between supply and network driven optimisation. A report to DECC,” no. November, pp. 1–104, 2010.

[2] G. Strbac, C. K. Gan, M. Aunedi, V. Stanojevic, P. Djapic, J. Dejvises, P. Mancarella, A. Hawkes, D. Pudjianto, D. Openshaw, S. Burns, P. West, D. Brogden, A. Creighton, and A. Claxton, “Benefits of Advanced Smart Metering for Demand Response based Control of Distribution Networks,” no. April 2010, p. 49, 2010.

[3] H. Saadat, *Power System Analysis*, Second Edi. Milwaukee: McGraw Hill, 2004.

[4] National Grid, “Balancing Services.” [Online]. Available: http://www2.nationalgrid.com/uk/services/balancing-services/. [Accessed: 10-Apr-2016].

[5] UK Power Networks, “Our Networks, Your Power: An Introduction to UK Power Networks,” 2012.

[6] Energy Research Partnership, “The future role for energy storage in the UK -- Main Report,” 2011.

[7] UK Power Networks, “Design and Planning considerations for large-scale distribution-connected energy storage (SNS1.2),” 2013.

[8] A. S. O. Ogunjuyigbe, C. G. Monyei, and T. R. Ayodele, “Price based demand side management: A persuasive smart energy management system for low/medium income earners,” *Sustain. Cities Soc.*, vol. 17, pp. 80–94, 2015.

[9] J. Schofield, V. Stanojevic, M. Bilton, G. Strbac, and J. Dragovic, “Application of demand side response and energy storage to enhance the utilization of the existing distribution network capacity,” *22nd Int. Conf. Exhib. Electr. Distrib. (CIRED 2013)*, no. 0852, pp. 0852–0852, 2013.

[10] Energy Research Partnership, “The future role for energy storage in the UK -- Main Report,” 2011.

[11] International Electrotechnical Commission, “Electrical Energy Storage White paper,” 2011.

[12] S. Gill, I. Kockar, and G. W. Ault, “Dynamic Optimal Power Flow for Active Distribution Networks,” *IEEE Trans. Power Syst.*, 2013.

[13] S. Merz, “Current Technologies Issues and Identification of Technical Opportunities for Active network Management (ANM),” *BERR Emerg. Energy Technol. Program.*, 2008.

[14] M. Rowe, W. Holderbaum, and B. Potter, “Control methodologies: Peak reduction algorithms for DNO owned storage devices on the Low Voltage network,” *2013 4th IEEE/PES Innov. Smart Grid Technol. Eur. ISGT Eur. 2013*, pp. 1–5, 2013.

[15] F. Milano, *Power system modelling and scripting*, First. Ciudad Real, Spain: Springer, 2010.

[16] E. Z. Zhou, “Object-oriented programming, C++ and power system simulation,” *IEEE Trans. Power Syst.*, vol. 11, no. 1, pp. 206–215, 1996.

[17] Dictionary.com, “Define Simulation,” *Dictionary.com*. [Online]. Available: http://www.dictionary.com/browse/simulation. [Accessed: 20-Apr-2016].

[18] J. Arrillaga and C. P. Arnold, *Computer Analysis of Power Systems*. Christchurch, New Zealand: John Wiley & Sons, 1990.

[19] I. Bojanova, J. Zhang, and J. Voas, “Cloud Computing,” *IEEE IT Prof.*, vol. 15, no. 2 - March / April, pp. 12–14, 2010.

[20] Datamation, “SaaS Market Growing by Leaps and Bounds: Gartner,” *IT Business Edge*, 2010. [Online]. Available: http://www.datamation.com/entdev/article.php/3895101/SaaS-Market-Growing-by-Leaps-and-Bounds-Gartner.htm.

[21] P. . Deitel and H. . Deitel, *Internet & World Wide Web: How to Program*, 4th ed. New Jersey: Pearson Education, 2008.

[22] The PHP Group, “What is PHP?,” *PHP Official Website*. [Online]. Available: http://www.php.net/manual/en/intro-whatis.php. [Accessed: 08-Oct-2015].

[23] Microsoft, “ASP.NET | The ASP.NET Site.” [Online]. Available: http://www.asp.net/. [Accessed: 12-Apr-2016].

[24] J. Byrne, C. Heavey, and P. J. Byrne, “A review of Web-based simulation and supporting tools,” *Simul. Model. Pract. Theory*, vol. 18, no. 3, pp. 253–276, Mar. 2010.

[25] IEEE Open Source Software Task Force and F. Milano, “IEEE Open Source Software.” [Online]. Available:

http://ewh.ieee.org/cmte/psace/CAMS_taskforce/software.htm. [Accessed: 10-Feb-2013].

[26] D. Powers, *PHP object-oriented solutions*. Berkeley, CA, United States of America: Apress, 2008.

[27] W3Techs: Web Technology Surveys, "Usage Statistics and Market Share of PHP for Websites, January 2014," 2014. [Online]. Available: http://w3techs.com/technologies/details/pl-php/all/all.

[28] W3Techs: Web Technology Surveys, "Usage of server-side programming languages for websites," *Web Technology Surveys*, 2014. [Online]. Available: http://w3techs.com/technologies/overview/programming_language/all.

[29] M. J. Dolan, S. Gill, C. Foote, G. W. Ault, G. Bell, and M. Barnacle, "Modelling and Delivery of an Active Network Management Scheme for the Northern Isles New Energy Solutions Project(Paper 1381)," in *22 nd International Conference on Electricity Distribution*, 2013, no. 1381.

[30] AES UK and Ireland, "Kilroot Advancion Energy Storage Array." [Online]. Available: http://aesukireland.com/our-business/energy-storage/kilroot-energy-storage/default.aspx. [Accessed: 14-Apr-2016].

[31] Office for Gas and Electricity Markets (OFGEM), "Feed-in Tariff (FIT) scheme," 2016. [Online]. Available: https://www.ofgem.gov.uk/environmental-programmes/feed-tariff-fit-scheme. [Accessed: 14-Apr-2016].

[32] Moixa Technology Limited, "Maslow Smart Energy Storage," 2016. [Online]. Available: http://www.meetmaslow.com/.

[33] H. Ibrahim, R. Beguenane, and A. Merabet, "Technical and financial benefits of electrical energy storage," *2012 IEEE Electr. Power Energy Conf. EPEC 2012*, pp. 86–91, 2012.

[34] G. W. Ault, S. Gill, and I. Kockar, "Using Dynamic Optimal Power Flow to Inform the Design and Operation of Active Network Management Schemes Paper 1327," in *22 nd International Conference on Electricity Distribution*, 2013, no. 1327.

[35] I. Papic, "Simulation Model for Discharging a Lead-Acid Battery Energy Storage System for Load Leveling," *IEEE Trans. Energy Convers.*, vol. 21, no. 2, pp. 608–615, 2006.

[36] A. Purvins, I. T. Papaioannou, and L. Debarberis, “Application of battery-based storage systems in household-demand smoothening in electricity-distribution grids,” *Energy Convers. Manag.*, vol. 65, pp. 272–284, Jan. 2013.

[37] British Broadcasting Corporation, “Britain From Above: Tea-time Britain, How the National Grid responds to demand,” British Broadcasting Corporation, United Kingdom, 2010.

[38] M. Rowe, T. Yunusov, S. Haben, C. Singleton, W. Holderbaum, and B. Potter, “A peak reduction scheduling algorithm for storage devices on the low voltage network,” *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 2115–2124, 2014.

[39] A. Gabash and P. Li, “Active-Reactive Optimal Power Flow in Distribution Networks With Embedded Generation and,” *IEEE Trans. Power Syst.*, vol. 27, no. 4, pp. 2026–2035, 2012.

[40] A. Gabash and P. Li, “Flexible optimal operation of battery storage systems for energy supply networks,” *IEEE Trans. Power Syst.*, vol. 28, no. 3, pp. 2788–2797, 2013.

[41] S. S. Skiena, *The Algorithm Design Manual*, 1st Editio., vol. 1, no. 11. London: Springer, 2008.

[42] T. Chu, J. Qin, and J. Wei, “Distributed Storage Operation in Distribution Network with Stochastic Renewable Generation,” in *IEEE Power Engineering Society General Meeting Conference & Exposition*, 2014, pp. 1–5.

[43] R. Sioshansi, S. H. Madaeni, and P. Denholm, “A dynamic programming approach to estimate the capacity value of energy storage,” *IEEE Trans. Power Syst.*, vol. 29, no. 1, pp. 395–403, 2014.

[44] D. Neves and C. A. Silva, “Optimal electricity dispatch on isolated mini-grids using a demand response strategy for thermal storage backup with genetic algorithms,” *Energy*, vol. 82, pp. 436–445, 2015.

[45] R. J. Kerestes, G. F. Reed, and A. R. Sparacino, “Economic analysis of grid level energy storage for the application of load leveling,” *IEEE Power Energy Soc. Gen. Meet.*, pp. 1–9, 2012.

[46] K. M. U. Ahmed, M. Ampatzis, S. M. Ieee, P. H. Nguyen, M. Ieee, W. L. Kling, and S. M. Ieee, “Application of Time-series and Artificial Neural Network Models in Short Term Load Forecasting for Scheduling of Storage Devices,” 2014.

[47] S. Shao, M. Pipattanasomporn, and S. Rahman, “Demand response as a load shaping tool in an intelligent grid with electric vehicles,” *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 624–631, 2011.

[48] P. Siano, “Demand response and smart grids - A survey,” *Renew. Sustain. Energy Rev.*, vol. 30, pp. 461–478, 2014.

[49] P. Siano, “Demand response and smart grids - A survey,” *Renew. Sustain. Energy Rev.*, vol. 30, pp. 461–478, 2014.

[50] L. Zhao and V. Aravinthan, “Strategies of residential peak shaving with integration of demand response and V2H,” *Asia-Pacific Power Energy Eng. Conf. APPEEC*, no. i, 2013.

[51] C. Heymans, S. B. Walker, S. B. Young, and M. Fowler, “Economic analysis of second use electric vehicle batteries for residential energy storage and load-levelling,” *Energy Policy*, vol. 71, pp. 22–30, 2014.

[52] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. Chichester, West Sussex: John Wiley & Sons, 1990.

[53] S. Martello and P. Toth, “Subset-sum problem,” in *Knapsack problems: Algorithms and computer interpretations*, 1st Editio., Chichester, West Sussex: John Wiley & Sons, 1990, pp. 105–130.

[54] E. W. Weisstein, “Bin-Packing Problem,” *MathWorld--A Wolfram Web Resource*. [Online]. Available: http://mathworld.wolfram.com/Bin-PackingProblem.html. [Accessed: 03-Mar-2015].

[55] S. Martello and P. Toth, “Bin-Packing problem,” in *Knapsack problems: Algorithms and computer implementations*, Chichester, New York: John Wiley & Sons, 1990, pp. 222–243.

[56] J. Malkevitch, “Bin Packing and Machine Scheduling,” *American Mathematical Society Feature Column*. [Online]. Available: http://www.ams.org/samplings/feature-column/fcarc-packings3. [Accessed: 03-Mar-2015].

[57] J. H. Holland, *Adaptation in Natural and Artificial Systems: An introductory analysis with applications to biology, control, and artificial intelligence.* Oxford, England: Michigan Press, 1975.

[58] D. Kleinjan, “Genes and their expression: Genetics in Modern Medicine,” in *Genes*

and Common Diseases: Genetics in Modern Medicine, A. Wright and N. Hastie, Eds. Cambridge, UK: Cambridge University Press, pp. 3–19.

[59] A. Shukla, H. M. Pandey, and D. Mehrotra, "Comparative Review of Selection Techniques in Genetic Algorithm," pp. 515–519, 2015.

[60] MathWorks, "Genetic Algorithm," *MATLAB R2014a Online Manual;*, 2014. [Online]. Available: http://uk.mathworks.com/discovery/genetic-algorithm.html?s_tid=gn_loc_drop. [Accessed: 20-Apr-2016].

[61] K. Tatroe, P. MacIntyre, and R. Lerdorf, *Programming PHP*, 3rd ed., no. August. Sebastobopol, California: O'Reilly, 2013.

[62] W3 Consortium, "Architecture of the World Wide Web, Volume One," *W3C Recommendation 15 December 2004*, 2004. [Online]. Available: https://www.w3.org/TR/webarch/. [Accessed: 09-Apr-2016].

[63] Mozilla Developer Network, "How the Web works," *Mozilla Developer Network*, 2015. [Online]. Available: https://developer.mozilla.org/en-US/Learn/Getting_started_with_the_web/How_the_Web_works. [Accessed: 09-Apr-2016].

[64] I. Bojanova and A. Samba, "Analysis of Cloud Computing Delivery Architecture Models," *2011 IEEE Work. Int. Conf. Adv. Inf. Netw. Appl. Biopolis, Singapore*, pp. 453–458, Mar. 2011.

[65] S. Chen and F. Lu, "Web-based simulations of power systems," *Comput. Appl. Power, IEEE*, no. January, pp. 35 – 40, 2002.

[66] Open Electrical, "Power Systems Analysis Software," 2013. [Online]. Available: http://www.openelectrical.org/wiki/index.php?title=Power_Systems_Analysis_Softwar e.

[67] R. Leou and Z. Gaing, "A Web-based load flow simulation of power systems," *IEEE Power Eng. Soc. Summer Meet. Chicago, IL, USA*, pp. 1587–1591, 2002.

[68] J. Yang, F. Lin, and Y. Fu, "Development of a Web-Based Software for Micro Power System Design," *2010 Int. Conf. Electr. Control Eng. Wuhan, China*, pp. 2748–2751, Jun. 2010.

[69] S. Tan and J. Yang, "Internet-based platform for power system simulating and planning," in *2011 Second International Conference on Mechanic Automation and*

Control Engineering, Inner Mongolia, China, 2011, pp. 2271–2274.

[70] BCP Busarello + Cott + Partners AG, “NEPLAN 360 Overview.” [Online]. Available: http://www.neplan.ch/html/e/e_PowerSystems_Properties_default_web.htm. [Accessed: 03-Mar-2013].

[71] F. Milano, “An open source power system analysis toolbox,” *IEEE Trans. Power Syst.*, vol. 20, no. 3, pp. 1199–1206, 2005.

[72] MathWorks, “Call MATLAB functions from a Web Application,” *MATLAB R2014a Online Manual; MATLAB COM Automation Server*. [Online]. Available: http://www.mathworks.co.uk/help/matlab/matlab_external/call-matlab-functions-from-a-web-application.html.

[73] InterPSS Community, “InterPSS 2.0 Manual and Documentation.” [Online]. Available: https://sites.google.com/a/interpss.org/interpss/Home/interpss-2-0.

[74] R. Zimmerman, C. Murillo-Sanchez, and D. Gan, “MATPOWER: A MATLAB Power System Simulation Package Documentation,” 2014. [Online]. Available: http://www.pserc.cornell.edu//matpower/.

[75] Neplan AG and BCP Switzerland, “NEPLAN 360 Overview.” [Online]. Available: http://www.neplan.ch/html/e/e_PowerSystems_Properties_default_web.htm.

[76] MathWorks, “How do I leverage the MATLAB Engine to deploy my MATLAB code over the web for an ASP.NET application using MATLAB 7.3 (R2006b),” *MATLAB Central: MATLAB Answers*, 2012. [Online]. Available: http://www.mathworks.com/matlabcentral/answers/99541-how-do-i-leverage-the-matlab-engine-to-deploy-my-matlab-code-over-the-web-for-an-asp-net-application.

[77] MathWorks, “MATLAB Compiler SDK,” 2014. [Online]. Available: http://uk.mathworks.com/products/matlab-compiler-sdk/features.html. [Accessed: 09-Apr-2016].

[78] H. Chen, C. Cañizares, and A. Singh, “Web-based Computing for Power System Applications,” *Proc. North Am. Power Symp. (NAPS), San Luis Obispo, California*, 1999.

[79] M. Alex, “PHP: a fractal of bad design,” *Fuzzy Notepad*, 2012. [Online]. Available: http://me.veekun.com/blog/2012/04/09/php-a-fractal-of-bad-design/.

[80] E. McGrath, “12 ‘Must-Know" Advantages of PHP,” *Vandelay Web Development*

blog, 2012. [Online]. Available: http://www.vandelaydesign.com/advantages-of-php/. [Accessed: 09-Oct-2015].

[81] European Commission Directorate-General for Research; Sustainable Energy Systems, “European SmartGrids Technology Platform,” Brussels, 2006.

[82] X. Lu, Z. Lu, and W. Wang, “On Network Performance Evaluation toward the Smart Grid: A Case Study of DNP3 over TCP/IP,” *2011 IEEE Glob. Telecommun. Conf. - GLOBECOM 2011, Houston, TX, USA*, pp. 1–6, Dec. 2011.

[83] R. Amiri and O. Elkeelany, “An embedded TCP/IP hard core for Smart Grid information and communication networks,” *Proc. 2012 44th Southeast. Symp. Syst. Theory (SSST),Jacksonville, Florida, USA*, pp. 185–189, Mar. 2012.

[84] H. Zhao, J. Evans, S. Tu, I. Proctor, M. Yang, X. Qi, M. Williams, Q. Gao, G. Ottoni, A. Paroski, and S. MacVicar, “The HipHop compiler for PHP,” *Proc. ACM Int. Conf. Object oriented Program. Syst. Lang. Appl. - OOPSLA ’12, Tucson, AZ, USA*, p. 575, 2012.

[85] Facebook, “HipHop Virtual Machine for PHP,” *HipHop Virtual Machine Specification*. [Online]. Available: https://github.com/facebook/hhvm/wiki.

[86] A. Crane, “Experiences of Using PHP in Large Websites,” *UK Unix and Open Systems User Group*. [Online]. Available: http://www.ukuug.org/events/linux2002/papers/html/php/index.html. [Accessed: 10-Oct-2015].

[87] The PHP Group, “Math:: PEAR Packages,” *PEAR Packages*. [Online]. Available: http://pear.php.net/packages.php?catpid=15&catname=Math. [Accessed: 10-Oct-2015].

[88] M. Zhou, *High Performance Computing in Power and Energy Systems: Distributed Parallel Power System Simulation*. Berlin: Springer, 2013.

[89] Nginx Community, “NginX.” [Online]. Available: http://wiki.nginx.org/Main.

[90] L. Eshkevari, F. Dos Santos, J. R. Cordy, and G. Antoniol, “Are PHP applications ready for Hack?,” in *Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on*, 2015, pp. 63–72.

[91] The PHP Group, “PHP Manual - Mathematical Functions.” [Online]. Available: http://us3.php.net/manual/en/math.installation.php.

[92] The PHP Group, "PHP Manual: Mathematical Extensions." [Online]. Available: http://us3.php.net/manual/en/refs.math.php. [Accessed: 08-Oct-2015].

[93] The PHP Group, "PEAR: PHP Extension and Application Repository." [Online]. Available: http://pear.php.net/. [Accessed: 08-Oct-2015].

[94] The PHP Group, "PHP Manual: Arrays," *The PHP Manual*. [Online]. Available: http://us3.php.net/manual/en/language.types.array.php. [Accessed: 08-Oct-2015].

[95] The PHP Group, "PHP Manual: Lapack Class." [Online]. Available: http://us3.php.net/manual/en/class.lapack.php. [Accessed: 08-Oct-2015].

[96] J. M. Castagnetto, "PEAR: Math_Complex," *PEAR Packages*, 2010. [Online]. Available: http://pear.php.net/package/Math_Complex/.

[97] J. M. Castagnetto, "PEAR: Math_Vector," *PEAR Packages*, 2010. [Online]. Available: http://pear.php.net/package/Math_Vector/docs/latest/li_Math_Vector.html.

[98] Joyent, "Node JS Official Documentation," 2011. [Online]. Available: http://nodejs.org. [Accessed: 02-Feb-2015].

[99] Energy Systems Research Unit University of Sthratclyde, "Case Study: Riverside Community, Stirling," 2007. [Online]. Available: http://www.esru.strath.ac.uk/EandE/Web_sites/06-07/Carbon_neutral/case_study_folder/case study_.htm. [Accessed: 02-Feb-2015].

[100] Stirling Council, "2011 Census Stirling; Community Council Area Profiles: Riverside, Stirling Council area and Scotland," Stirling, Scotland, 2011.

[101] Ambri, "Ambri Liquid Metal Battery 2013 Progress Update," Cambridge, MA, 2013.

[102] Institute of Systems and Robotics University of Coimbra, "Residential Monitoring to Decrease Energy Use and Carbon Emissions in Europe," 2008. [Online]. Available: http://remodece.isr.uc.pt/. [Accessed: 10-Feb-2015].

[103] UK Power Networks, R. Cordwell, and M. Adolphus, "Engineering Design Standard EDS 08-119:Guidance for the application of ena er p2/6 security of supply," pp. 1–30, 2014.

[104] General Motors, "Chevrolet Volt Battery." [Online]. Available: https://media.gm.com/content/dam/Media/microsites/product/volt/docs/battery_101.pdf. [Accessed: 11-Feb-2016].

[105] The Apache Software Foundation, “Apache - HTTP Server Project.” [Online]. Available: http://httpd.apache.org/. [Accessed: 12-Oct-2015].

[106] The PHP Group, “PHP - Shell_Exec,” *PHP Official Website*. [Online]. Available: http://php.net/manual/en/function.shell-exec.php. [Accessed: 20-Oct-2015].

[107] The PHP Group, “PHP - Exec.” [Online]. Available: http://php.net/manual/en/function.exec.php. [Accessed: 08-Oct-2015].