Optimising Charging of Electric Vehicles through the use of Genetic Algorithms

Tom Curran

Supervisor: Dr. Edgar Galván-López

Co-Supervisor: Prof. Stephen Barrett

A dissertation submitted to the University of Dublin, in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

2014

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: _____

Tom Curran

29/08/2014

Permission to Lend and/or Copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: _____

Tom Curran

29/08/2014

Abstract

The charging of electric vehicles (EVs) places a strain on the transformers in the electric grid in order for it to be able to supply enough power to meet demand. This is set to become more and more of a burden on the grid as EVs become more popular among the general public. The use of demand-side management (DSM) and smart grids have been investigated as a solution to this problem, and the problem of reducing load at peak times in general, by automatically shifting the use of some household appliances to times where there is a reduced load.

This dissertation aims to reduce these issues in a number of ways through the use of genetic algorithms (GAs) alongside the DSM approach. The proposed approach is to implement a GA which uses several 'fitness functions' in order to target multiple separate objectives; fitness functions can be seen as black boxes which we can use to indicate how good/bad a potential solution is. The objectives include a lower overall transformer load, a lower peak-to-average ratio, a final state of charge (SoC) close to the normal approach, and a lower cost of electricity to consumers. The execution of this GA will create schedules that the EVs can follow throughout each day in order to charge whilst achieving these goals; in order to evaluate this implementation, we will simulate the electric grid using GridLAB-D across a 28-day period (for each approach) in which the EVs arrive home in the evening with a generated SoC, and must charge to the target SoC by the time of their departure in the morning.

Acknowledgements

I would first like to thank my supervisor, Dr. Edgar Galván-López, who directed me through this work and guided its progression throughout the past six months.
I would also like to thank my partner Siobhán, who helped keep me going through the past year with constant reassurance and unlimited patience.
I also wish to acknowledge my friends Kieran and Luke, who have accompanied me through every difficult deadline through the past five years at Trinity.
Finally, I would like to express my appreciation to my family. My brother Niall, and my parents, Paul and Susan, who have given unending support throughout my education, and without whom I would never have achieved what I have.

For my parents, Paul and Susan, who have supported me through every endeavour.

Contents

Contents **vi**

1 Introduction **1**

2 State of the Art **3**

2.1 The Electric Grid 3

2.2 Demand-Side Management 4

2.3 Algorithmic Approaches 5

2.4 Summary 6

3 Problem Under Investigation **7**

3.1 Demand-Side Management 7

3.2 Transformer Load 8

3.3 Peak-to-Average Ratio 9

3.4 Electricity Cost 9

3.5 Summary 9

4 Genetic Algorithms **10**

4.1 Overview of Genetic Algorithms 10

4.2 Operation of Genetic Algorithms 11

5 Implementation **13**

5.1 Approach 13

5.2 Technical Details 14

5.2.1 GridLAB-D 14

5.2.2 Python & Pyevolve 15

5.2.3 MATLAB 16
5.3 Fitness Functions 16
5.3.1 Steady Charging Fitness Function 17
5.3.2 Standard Deviation Fitness Function 18
5.3.3 Pricing Signal Fitness Function 19
5.4 Experimental Setup 21

6 Experimental Results **24**
6.1 Transformer Load 24
6.2 State of Charge 28
6.3 Peak-to-Average Ratio 33
6.4 Cost of Electricity 35
6.5 Growth of fitness within the GAs 37
6.6 Summary 40

7 Conclusions/Future Work **41**
7.1 Conclusions from the Experimental Results 41
7.1.1 Steady Charging Fitness Function 41
7.1.2 Standard Deviation Fitness Function 42
7.1.3 Pricing Signal Fitness Function 42
7.1.4 Greedy Approach 43
7.2 Future Work 43
7.2.1 Real-Valued Encoding 43
7.2.2 Use of Real-World Models within GridLAB-D 44
7.2.3 Higher Numbers of EVs & The Use of Energy Storage Units 45

References **46**

Appendix **52**

Chapter 1

Introduction

Demand-side management (DSM) refers systems that allow the automatic control of energy demand from the consumers' side, implemented by the utility companies. This is an important concept within the domain of smart grids (SGs); it allows utility companies to establish control over the peak load, as well as giving power to balance the load through peak and off-peak cycles. Without this capability, the peak load must be achieved through the use of 'peaking plants' run only to meet peak demand, as well as causing an extra strain on transformers within the grid. One of the predominant ways in which DSM is used to balance load lies in load-shifting through the control of consumer appliances by the suppliers; this relies on the handing over of some control of certain household appliances from the consumer to the supplier, such that, at peak times, the supplier would be able to deactivate these appliances to reduce the peak and reactivate them when demand is low. The appliances that are time-independent enough to suit this approach include washing machines, dryers, and electric vehicle (EV) chargers.
EVs are an important appliance to consider when investigating within this domain; as EVs have become more popular among the general population in recent years, they bring an increased demand on the electric grid. As such, it is important to be able to take advantage of the time-independence of the charging of an EV in order to combat this potentially disruptive load. Many approaches have been investigated in the implementation of automated DSM systems, including Monte Carlo Tree Search, game theory-based methods, multi-agent systems, and evolutionary algorithms. This study aims to tackle the problems outlined through

the use of genetic algorithms (GAs), a stochastic bio-inspired search algorithm, whose typical representation (binary-encoded) is conducive to the on-off states of the EV chargers to be simulated.

The approach to be taken involves the development of a GA with a number of 'fitness functions' that will tackle the goals to be achieved; including reduced peak-to-average ratio (PAR), reduced overall load, and finally, a reduced cost to the consumers, all whilst aiming to achieve a similar final state of charge (SoC) for each of the EVs simulated. To ensure the scalability of the approach, the simulations will be run using different numbers of EVs, and our GA-based approach will be contrasted with the outcome of a standard grid in the same scenario. The execution of the GA will create schedules that the EVs can follow throughout each day in order to charge whilst achieving the goals set; in order to evaluate this implementation, we will simulate the electric grid across a 28-day period (for each approach) in which the EVs arrive home in the evening with a given SoC, and must charge to the target SoC by the time of their departure in the morning.

In Chapter 2, we will discuss some of the literature representing the current state of the art in this domain, as well as uncover potential missing links in the literature which can be filled through this work. We will then move on to Chapter 3 to discuss the problem that we aim to solve, before briefly giving an overview of GAs and their operation in Chapter 4. Chapter 5 will discuss the implementation details of the proposed approach, as well as the experimental parameters that will be used to evaluate the system. Chapter 6 will present and discuss the results of the many experiments executed, before we discuss the conclusions made from this work, and the potential areas for future research in Chapter 7.

Chapter 2

State of the Art

2.1 The Electric Grid

There are many issues involved in the running of the traditional electric grid. The management of this system is only possible through outdated monitoring methods, and an ability to increase supply quickly to meet the demand at peak times; in many cases through the use of power plants operated solely at peak times, known as 'Peaking Plants' [1, 2, 3, 4]. There are a number of problems with this approach, including the high potential of peaks and troughs in the load profile despite attempts to balance from the supply side, unpredictability of demand, and the degradation of transformers in the grid due to the fluctuating load on them.

The Smart Grid (SG) is an electric grid with a means of using sensors, control technologies, and communication (over a LAN-like system [5]) in order to deliver electricity through the grid in a more effective and efficient manner [6]. This can be implemented throughout the grid from the supplier to consumers through the use of both hardware and software. In many cases it is seen that the move to smart grids will be necessary to achieve a number of essential goals, including distributed generation of electricity (such as home solar panels, wind turbines, etc), better demand response, and support of electric vehicles (EVs) as they become more prevalent [5, 7]. A key area of research for SGs is in the control of the demand from the consumers' side.

2.2 Demand-Side Management

One of the most explored areas in SGs is Demand-Side Management (DSM) systems, this is apparent when we look to the increasing number of related publications over the years [5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. DSM is a very important concept in SGs, it is concerned with the automatic control of energy demand from the consumers' side, implemented by the utility companies.
There are a number of methods of DSM discussed within the literature; for example, smart pricing, which relies on the installation of intelligent meters on the consumer's side, would allow the metering and analysis of electricity demand by the supplier. This could lead to control of pricing at a finer level than current systems, allowing energy companies to penalise/reward consumers for using/avoiding the use of electricity at times of high demand [5, 10]. Another method is direct load control (DLC), which involves the handing over of some control of certain household appliances from the consumer to the supplier, such that, at peak times, the supplier would be able to deactivate these appliances to reduce the peak and reactivate them at times of low load [5]. Examples of appliances suitable for this approach would be dryers, washing machines, or EV chargers, normally known as shiftable loads. There are many investigations into the concept of 'load-shifting' within this domain, and while many use limited dynamic/linear programming approaches, there are a large number that implement different approaches that show potential to scale to the larger number of different types of device on the grid [5, 12, 18].
There are a number of tools commonly used in simulating these grid systems, some of the more frequently seen open source tools include WindMil (Milsoft Utility Solutions, Inc.), GridLAB-D (US DoE), and OpenDSS (EPRI) [19]. However, due to its flexibility and ease of modification, as well as support for time-series data and scheduling of appliances, GridLAB-D is an excellent candidate for load-shifting in DSM and the simulation of experiments. A key reason for using GridLAB-D in this work was due to the fact that many studies report its flexibility in conducting experiments [11, 20, 21]. Another important tool used

consistently in the literature is MATLAB Simulink which allows the specification of models and analysis of simulation results.

2.3 Algorithmic Approaches

Intelligent approaches include the use of Monte Carlo Tree Search (MCTS) [12, 21], evolutionary algorithms (EAs) [11], multi-agent systems [10], as well as game theory-based methods [5, 18]. While these investigations take a different approach toward the implementation, the goals remain as a reduction in system peak load demand, load profile reshaping, and, to a lesser extent, a reduction in cost of electricity used. The reasons for these approaches stem from their ability to handle many different types of device on the grid, as well as flexibility to be used with different structures; they are also able to scale well to the sheer number of potential solutions when scheduling large numbers of different devices throughout a specified time period.

While the MCTS and game-theoretic approaches to DSM have shown good results in many of the targets proposed [5, 12], evolutionary algorithms have also shown the potential to scale to larger numbers of devices whilst only increasing the size of the chromosome encoding, as well as to different types of device with the same algorithm, a level of flexibility not afforded by other approaches. The use of binary GAs has been implemented within the literature to simulate the on/off capabilities of many household appliances, but there also lies the possibility of using real-valued encoding in the case of EV chargers, in order to charge at a fraction of the maximum power input in order for the SG to have higher precision control over the demand.

A key area within the Smart Grid space is the charging of EVs; much of the literature does not take a focus toward this issue, but there are examples where EVs are analysed as a shiftable load within the grid [11, 21]. These examples do however take a limited approach with regards to the number of EVs, with 9 and 6 EVs simulated respectively, as well as using a relatively high initial SoC in the former. In contrast, this work aims at testing the proposed approach using dozens of EVs with a relatively low initial SoC. EVs are an important area to investigate, due to their recent growth in popularity, and the impact that widespread adoption

of EVs would have on the grid, as during charging, an EV can almost double the average household load [5].

2.4 Summary

There are several gaps in the literature that we will aim to address. We see that there is potential in the use of genetic algorithms within this space, and while there have been investigations into the use of GAs in DSM, we can see scope to further emphasise the use of EVs on the grid, due to their growing popularity, and the potential impact this would have on the current grid structure. As such, we aim to simulate scenarios with larger numbers of EVs than have been investigated previously, as well as simulating with multiple different numbers of EVs, to gain insight into the scalability of the proposed solution. There is also scope to analyse an SG under the load of different types of EV (with differing battery capacities and charger power for example) to further investigate the potential outcome of multiple consumer EV types on the grid.
We also see an area of research in targeting multiple objectives, such as a lower peak load, PAR, and cost within the same systems, so as to show the possibility of having multiple goals achieved, potentially with a trade off between them that could be specified depending on the demand or priorities of the consumers/suppliers.
In the area of GAs, the literature analysed does not go into the detail of investigating fitness functions with different goals, which could show the areas in which a GA would be able to optimise better general solutions. We can thus aim to implement a GA which targets multiple objectives and can compare and contrast the outcomes of each of these using the commonly seen metrics from the literature (load profile, PAR, cost).

Chapter 3

Problem Under Investigation

There are a number of points of focus within this dissertation; the predominant problem, however, relates to the alleviation and balancing of the load on transformers within the electric grid. We will be observing and aiming to combat this problem by controlling the charging of electric vehicles (EVs) within the grid. The work investigated in this dissertation will assume the existence of either an implementation of a Smart Grid (SG) system, or a means of networked communication between the electric vehicles.

3.1 Demand-Side Management

The standard electric grid lacks any computation power or control over its own operation; this is not the case within an SG. Demand-side management (DSM) is an important area within the domain of SGs. The concept of DSM revolves around the control of energy consumption from the consumers' side, as opposed to the normal management from the suppliers' side. The goals in this practice for the energy supplier are typically to reduce the peak load, and reshape the load throughout peak and off-peak times in order to reduce the amount of fluctuation. The aspect of DSM that we will be focusing on is in 'load-shifting'; this takes advantage of the flexibility of operation of certain appliances at the consumers' side, such as washing machines, dryers, and most importantly, EV

chargers. These devices, known as 'shiftable' appliances, need only to complete their operation within a certain time frame; in our focus, EV chargers, this is typically simply that the EV needs only to be charged by the departure time of the user, with minimal need for the EV to have a high state of charge (SoC) in the interim. DSM operates in this manner by activating/deactivating shiftable appliances when the load demand gets high/low, allowing for the leveling out of the load profile, and reducing the strain on transformers within the grid. The standard grid differs heavily from this in that the EVs would simply charge once plugged in until they reach 100% SoC, adding to the already high load during peak hours.

3.2 Transformer Load

The level and stability of transformer load within the grid can impact it greatly. At peak times we see spikes in the transformer load, while during off-peak times, the load can drop dramatically; these fluctuations put a strain on the transformers and can degrade power quality, cause voltage problems, and damage to equipment on the grid [5]. In many cases, we also must resort to the use of 'peaking plants' which are operated only at peak times to alleviate the load. The control of this load profile and reduction in peak demand could increase grid sustainability due to the reduced overall cost and carbon emissions from reduced numbers of peaking plants. This issue looks to become a bigger issue due to the increasing consumer demand for plug-in EVs in recent years as they have become more viable for everyday use; this is due to the amount of power used in charging these EVs, as during charging time, an EV can almost double the average household load [5]. In this situation, DSM will become a necessity to continue the management of the electric grid; this dissertation looks toward creating an autonomous DSM to tackle this problem under the strain of EVs by using stochastic bio-inspired search.

3.3 Peak-to-Average Ratio

Peak-to-average ratio (PAR) is calculated as the load at the highest peak in the specified time period, divided by the average load across said time period. It is a useful metric in measuring the amount of fluctuation in the transformer load, but has the drawback of not representing the level of the load. For example, if a system remains relatively stable at a high load, it will have a low PAR value, close to 1.0 (optimal), while a system in which the load begins high but drops dramatically, we will see a high PAR, despite the potential that almost all of the load on the transformers is at the lower level. As such, while we will be analysing the PAR throughout this study, we will also be analysing many other factors to ensure a complete picture of the system.

3.4 Electricity Cost

The final aspect that we will be investigating is within the reduction of costs to the consumer for the electricity used in charging their EV. Electricity companies provide a number of price plans which give different rates per unit (kilowatt-hour, kWh) depending on the time of electricity usage. As such, moving non-essential energy usage from a peak time to an off-peak time reduces the cost to the consumer despite using the same amount of electricity. In this dissertation we look toward using DSM to load-shift in a number of ways, including load shifting to reduce the overall cost of electricity used to the users.

3.5 Summary

In summary, this dissertation aims to tackle the problems of high transformer load, high PAR, and high cost of electricity within the grid. The focus will be on the charging of EVs in a grid that will be simulated using the open source simulator GridLAB-D, as mentioned in the previous chapter. The solutions to the simulated scenarios will be developed using a stochastic bio-inspired search. We will discuss the algorithm for performing this search in the next chapter, before moving on to the implementation details in Chapter 5.

Chapter 4

Genetic Algorithms

4.1 Overview of Genetic Algorithms

Evolutionary Algorithms (EAs), a.k.a. Evolutionary Computation systems, are strongly influenced by the theory of evolution by natural selection. These algorithms have been with us for some decades and are very popular, perhaps, due to their successful application in a range of different problems, ranging from the automated design of circuits [22, 23], to the automated optimisation of game controllers [24].

Different research areas have been explored within EA (e.g., problem difficulty [25, 26, 27, 28, 29, 30, 31], semantics [32, 33]) . In this dissertation, however, we focus our attention on their applicability. In particular, we focus our attention on the use of Genetic Algorithms (GAs) due their representation (bitstring) which is suitable for the problem used in this work (see Chapter 3).

GAs utilise several key components from the theory of evolution, such as natural selection and mutation. A potential solution is defined within GAs as an individual; the group of these potential solutions is the population. GAs also rely on crossover (reproduction) between individuals in order to converge on an optimal solution to the defined problem. The algorithms move through generations of individuals, evaluating their 'fitness' and executing mutation and crossover logic in order to grow towards a fitter population.

4.2 Operation of Genetic Algorithms

In order to implement a GA, there are a number of key considerations. We must define the structure of the individuals; i.e., the potential solutions to our problem. In this dissertation, the solutions are schedules for the EVs to follow; as such, the individuals are 2D binary arrays where each entry is an on/off state for a specific EV (row) at a specific time (column). In simpler implementations, it could be a 1D binary string of a set length. In the following example, we will use this individual representation. Another key component is the fitness function; this can be considered as a black box into which we pass individuals. The fitness function will then return a value that grades the fitness of that individual, based on how good/bad a solution it is to the problem at hand. The fitness function is the most important part to the effective operation of the GA, as it is what allows the GA to converge on an optimal solution. There are then many different means of mutation and crossover that can be implemented in a GA; we will look at two such algorithms in the following example.
In this example we are looking to find a 1D binary string, of length 8, that contains only 1s; this is known as the OneMax problem. In this situation, we define the individuals to be an 8-digit binary string, and can allow the generation of the set population size to randomly create that number of strings. The fitness function in this situation is very simple, we can just return the number of 1s in the string that has been input to the function, as illustrated in Figure 4.1.

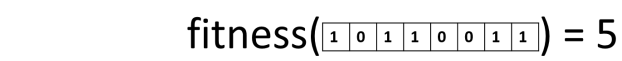

Figure 4.1: An illustration of the OneMax fitness function operating on an 8-digit string.

Figure 4.2 illustrates a mutation operator for the OneMax problem; in this case, we simply set a mutation rate which will act as the probability of mutation occurring in each bit of the individual, with the result of the operator being a swapping of the selected bit. Selection is then used to decide which individuals will undergo crossover within this generation; this can be done by choosing a random group (of a predefined size) of individuals in the population, and performing

crossover on the two fittest individuals. Crossover within this problem is also a simplistic operation, in which two individuals that have been selected, are split at a randomly selected index, and combined with each others opposing parts, resulting in two new individuals for the next generation.

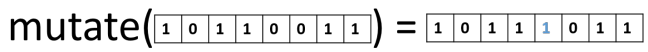

Figure 4.2: An illustration of the OneMax mutation operator operating on an 8-digit string, where the point of crossover has been chosen as the midpoint of the individual.

Figure 4.3: An illustration of the OneMax crossover function operating on an 8-digit string.

The final concept within GAs that was undertaken in the implementation of this study is that of 'Elitism'. Elitism involves the retention of the fittest individual of each generation, to ensure that there is no regression between generations in the best fitness. The result of this practice is that as the generations progress, the best fitness of each generation can only increase or remain the same, improving the success rate of the algorithm in converging on an optimum solution.
As stated at the beginning of this chapter, GAs were chosen for this study due their representation (bitstring) which is suitable for the problem used in this work. As well as this GAs have the benefit of scaling to larger numbers of devices whilst only increasing the size of the chromosome encoding, as well as to different types of device with the same algorithm. This flexibility is not available through alternative approaches.

Chapter 5

Implementation

5.1 Approach

The implementation approach to this experiment is to simulate an electric grid, under which we will be simulating a set number of EVs. We will then analyse the transformer load, peak-to-average ratio (PAR) values, final state of charge (SoC) of the EVs, and the cost of electricity used within these simulations. The simulations will be run for each of the algorithms developed so that comparisons can be made between all approaches under the same simulation parameters. The simulations will also be run multiple times using different numbers of houses with EVs to show how each approach scales to different grid sizes.

The method by which the algorithm will operate will be by designing a schedule for the EVs to follow throughout the simulation; the aim is for these schedules to provide improvements in the goals that each algorithm sets out to achieve. The schedules themselves will be a set of instructions for the EVs to follow throughout the night, with a decision to be made at every time slot during the simulated time period. The decisions available to the EV will be to either charge or not charge for that time period; as such, we will use schedules with a structure of 2D matrices of binary values with each column representing a time slot in the simulated time period, and each row representing an EV in the simulation. The GA will operate by creating a population of individuals (randomly generated), with each individual being a potential schedule to use. The GA will then evolve through the set

number of generations, improving the solutions as it progresses until reaching the final generation, at which point it will output the fittest schedule as the one to be used for that day. This process is repeated for each day within the simulation, for each different fitness function implemented, and for each number of EVs used in the simulation.

5.2 Technical Details

In this section we will discuss in detail some of the technical details of the implementation, looking toward the technologies used and what modifications were required, as well as some of the technical issues experienced and downsides to the technologies used.

5.2.1 GridLAB-D

GridLAB-D is a "power distribution system simulation and analysis tool"; it was developed by the United States Department of Energy in collaboration with industry and academia [34]. GridLAB-D provides a highly-customisable set of modules which can simulate a huge number of household appliances, including EV chargers. It also allows for further customisation through the integration of new modules, but this functionality was unnecessary during the implementation of this dissertation, as a modified version of the EV charger module provided all needed functionality. The operation of GridLAB-D relies on the input of 'Models', which specify the structure of the electric grid to be simulated, along with the number of EVs, houses, and the types of appliances within each household. These models can be specified to a high degree of accuracy, including using defined schedules for appliances and lights, which can be altered depending on the time of year. This could be used to implement the simulation of the approaches implemented through both the Summertime and Wintertime electricity pricing schedules.
Modifications to the included modules and the system itself are possible due to

the open-source nature of GridLAB-D. It is a relatively straightforward process to download the source version of the system, make modifications where needed, and recompile an altered version of the simulation tool. This was invaluable during the implementation stage of this dissertation, and the system's flexibility was the main reason for choosing it over alternatives, as it allowed the modification of the EV charger module in a number of ways. While the light systems in GridLAB-D allow the specification of schedules, many other modules do not; notably, the EV charger module. The experiments of this dissertation use schedules output by the GAs implemented, so a modification needed to be made to the EV charger to be able to read these output schedules and act accordingly throughout the simulation. There also needed to be alterations to allow a higher EV charger power, an increase from the standard plug-in charging 1700W to 3300W for some of the simulations.

5.2.2 Python & Pyevolve

Python was chosen for the GA implementation in this dissertation, as well as the general purpose scripting done to ensure all output was merged together and that the results were in a format easily interpreted for graphing. There were three main reasons why this language was chosen. Firstly, due to the simple syntax, interpreted nature of program operation, and simple installation process, it is very straightforward and fast to develop programs and scripts when compared to other languages; this allowed for rapid development and testing of the algorithms. Secondly, Python provides powerful integration with the operating system, making it very straightforward to develop the scripts needed to ensure all output was consistent, multiple sets of results were compiled into readable CSV files, and the files were available in a single place. However, some 'Bash' scripts were also created to automate the entire compilation process of GridLAB-D, along with the running of all Python programs in succession to generate and compile all data together and plot the results.
The final reason for choosing Python was because there are a huge number of libraries available to Python that allow for simple manipulation of data and output,

which can increase the speed of development further due to the instant availability of commonly-used logic. One of the key libraries used in this dissertation was Pyevolve, a genetic algorithm framework for Python [35], which provides many of the commonly-used operators in genetic algorithms, allowing us to explore the numerous approaches undertaken.

5.2.3 MATLAB

MATLAB was used for the final compilation of results for all simulations, as well as the plotting of all graphs. The output from GridLAB-D, along with all status output from the GAs were compiled together in CSV files by the Python programs implemented, this was then easily read by MATLAB and operated upon. MATLAB scripts were developed that could be run from command line to generate the tables and graphs needed for the results in this dissertation, these commands were then integrated into the Bash scripts mentioned previously, such that the entire operation, from generating models for GridLAB-D, through the running of GridLAB-D for all scenarios and algorithms, to the compilation/plotting of results in MATLAB, were automated through the execution of a single Bash script. Thousands of experiments were run in the investigation of the functions implemented, as such, MATLAB was a necessary tool in compiling and plotting results.

5.3 Fitness Functions

Three fitness functions were implemented for this dissertation; as such, the GA was run three times for each simulation, once with each of the fitness functions, to generate three sets of schedules which would then be used in GridLAB-D. These fitness functions are integral parts of the GA, they define the fitness evaluations of individuals for the GA to be able to converge on a solution that is overall 'better' than in any previous generation.

The fitness functions operate on individuals passed in as argument, they analyse the individual and return a score which will be used by the GA to compare to

other individuals. In these experiments, the individuals are the potential schedules to be used by the EVs each day of the simulation. The representation in this regard is a 2D matrix of binary values, with each row representing an EV, and each column representing a time slot in the simulation. The value at each point symbolises whether the EV (that row) should be on or off at that time slot (column). In order to get a full picture of the simulation, the fitness functions are also able to read the model generated, and see what initial SoC each EV has, as well as the capacity of each battery to infer how much power would be required to charge them. As well as this, the third fitness function is also able to read pricing information for the price plan in use. The fitness functions can then evaluate each individual based on overall transformer load on the grid at any time, as well as the amount of charge each car gets, and when during the night the power is supplied.
The GAs themselves share a configuration file which details all of the parameters for the operations of the GA. This parameters file includes all of the parameters detailed in Table 5.2 in the 'Experimental Setup' section of this document.

5.3.1 Steady Charging Fitness Function

The operation of this fitness function is the most straightforward of the three fitness functions implemented. The GA reads in details about the EVs in the simulation from the model, including initial SoC for each car each day, and the capacity of the batteries. From this, it can calculate how many time slots each EV will need to be able to achieve a specified charge value (for the experiments run for this dissertation, the target is 100% final SoC). The score is then calculated based on how close each EV is to this charge level after all time slots, with a score of 1.0 for each EV being the maximum score attainable. The score awarded per EV is based on a calculation of the percentage final SoC achieved, with the score being extracted as a linear function of the SoC value; the line ranges from a score of 0.0 at the base percentage possible, to a score of 1.0 at the optimum final SoC, the score then drops from that point to 0.0 again at the final SoC specified. Figure 5.1 illustrates the function under the experimental

parameters given, where the base percentage is 40% and the upper boundary is at 120%, at which point the solution would get 0.0 for that EV; going beyond this point would give a negative value for this row (EV) in the solution. This shape was chosen to encourage a general growth toward the optimal solution, with a quick drop in fitness for EVs that take more than the target amount of electricity during the schedule. The specification of a sharper penalty in score for an EV

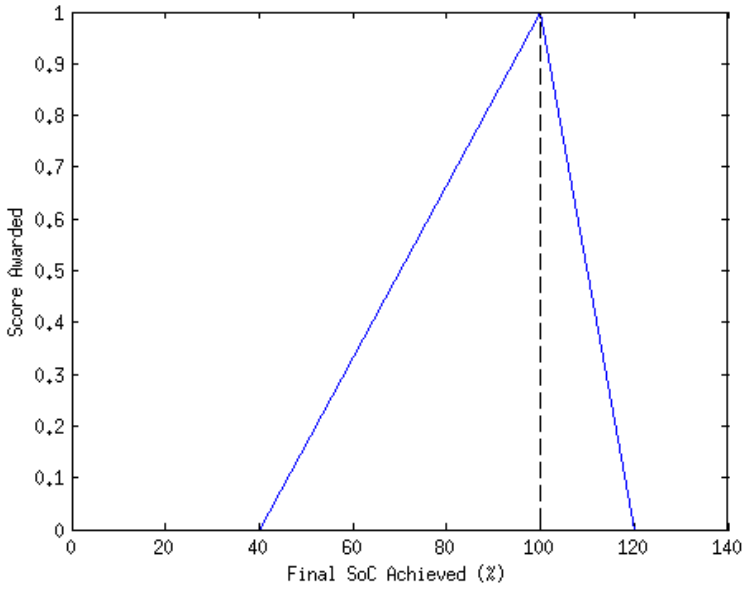

Figure 5.1: Plot of fitness score awarded per EV based on final SoC achieved.

going above its needed charge ensures that the load on the transformers remains at the lowest level it can be overall, whilst still achieving the specified final SoC target for each EV in the system.

5.3.2 Standard Deviation Fitness Function

This fitness function works in a similar manner to the steady charging fitness function (illustrated in Figure 5.1) for a large portion of its execution; however, once it reaches an average score of 0.8 per EV under the conditions of the previous fitness function, it then is able to supplement the fitness score with another

calculation. This is the point in which the fitness function analyses the standard deviation of the load on the transformers in the system. This is calculated by totaling all of the columns in the individual, showing the total number of EVs charging at each time slot in the simulation; we are aiming to keep this as steady as possible, so the target is a standard deviation of 0.0 for these values. Using this analysis, the fitness function can award a further score of 0.5 per EV for an optimal standard deviation value, giving a new optimum solution score of 1.5 per EV, with the score value being multiplied by the number of EVs to ensure a representative score for the individual. The score awarded for standard deviation is once again plotted on a line, in this case from 0.5 for an optimal standard deviation of 0.0, to a score of 0.0 for the empirically calculated standard deviation of 5.0. This line plot is illustrated in Figure 5.2.

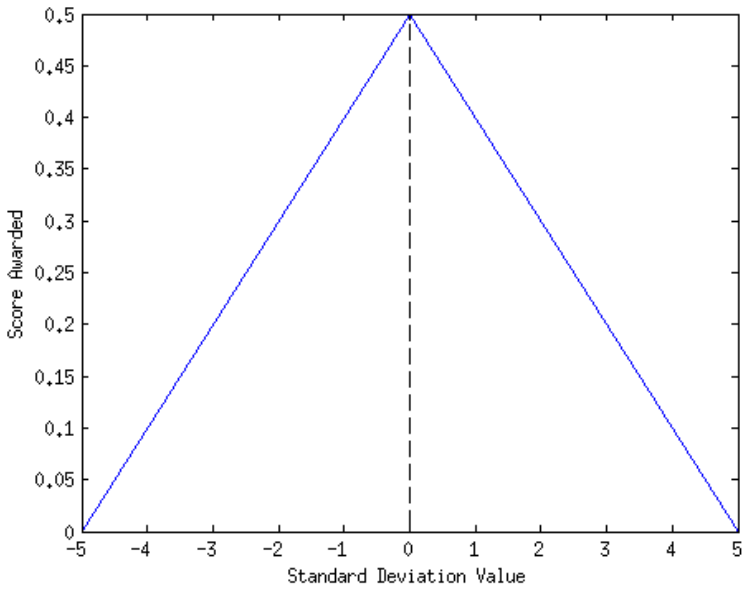

Figure 5.2: Plot of fitness score awarded per EV based on standard deviation value calculated across load on transformers.

5.3.3 Pricing Signal Fitness Function

The final fitness function developed targets a low cost to the consumer for the electricity used during the simulated time period. To achieve this, the fitness

function reads in the pricing signal data provided, which gives the price per kWh of electricity at every time slot throughout the simulated day. This fitness function works in three stages, the first stage operates exactly as the second fitness function does, where it runs the logic of the steady charging fitness function until achieving an average 0.8 score per EV; after this, the fitness function evaluates the cost of the electricity used. To do this, we once again plot the score to award against the targets to achieve. To plot this, we must set minimum and maximum prices to award scores of 0.0 and the maximum score of 0.7. The minimum price is calculated as the cost of electricity if all necessary units were used to charge all EVs at the off-peak electricity rate; the maximum rate is this number of units at the peak rate. Figure 5.3 shows this plot from minimum to maximum price going from a score of 0.7 to a score of 0.0. The score attained is multiplied by the number of EVs to give a representative score for the individual overall based on the cost. Upon achieving an average of 0.5 per EV out of the maximum 0.7,

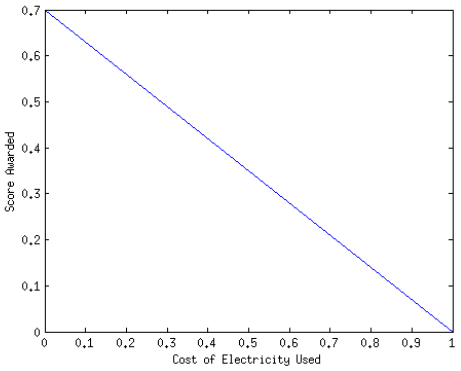

Figure 5.3: Plot of fitness score awarded for the individual based on the total cost of electricity used to the consumers in the simulation. This simplified plot shows the score awarded from 0.0 (the minimal cost possible) to 1.0 (the maximum cost possible).

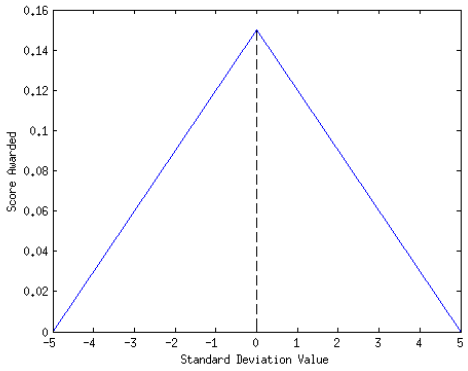

Figure 5.4: Plot of fitness score awarded for the individual based on the standard deviation of the transformer load on either side of the price rate increase.

the fitness function moves onto the third and final stage, in which it calculates and rewards a balanced transformer load on either side of the electricity rate switchover. We expect to see a spike in transformer load at this point as this

would reduce the cost overall, so our aim is to reduce the PAR by balancing either side as much as possible. The logic for this works in a similar way to the second fitness function, except that the plots are split between either side of the price change, which the fitness function can see from the pricing signal file. Once the values have been split at this point, each side works the same way as the standard deviation fitness function, with a maximum score of 0.15 per EV on either side, giving a total maximum score of 0.3 per EV; each of these are once again multiplied by the number of EVs to give the final score. These identical plots are illustrated in Figure 5.4.
The sum of these scores is then used as the score for the individual, giving a maximum score of 2.0 per EV for each individual under this fitness function.

5.4 Experimental Setup

Throughout the experiments, a number of constants were maintained to ensure fair comparisons between results. The simulated EV batteries had a capacity of 24kWh, based on the standard battery in a Nissan Leaf, which is one of the more popular EVs in Ireland currently. The experiments were run with the default 1700W charger used in GridLAB-D; however, to ensure scalable operation of the algorithms, the experiments were also run with 3300W chargers, the standard charger of a Nissan Leaf; the plots for these tests are available in the Appendix. The simulations are run from the cars arriving home at 18:00, until 07:30 when they leave. The initial SoC upon arriving home ranges from 40-50%, and the experiments target a final SoC of 100% by 07:30. The pricing signal used is the Electric Ireland 'Standard Electricity NightSaver' plan[1] on the Summertime schedule; on this plan currently, the peak time is from 9am until midnight, with a rate of €0.2062 per kWh, and the off-peak period is from midnight until 9am, with a rate of €0.1019 per kWh. These experimental parameters are summarised in Table 5.1.
There were also a number of constant parameters used in the operation of the GA under all fitness functions to ensure directly comparable results. These values are illustrated in Table 5.2.

[1]Source: https://www.electricireland.ie/switchchange/allPricePlans.htm

Parameter	Value
Charger Input	1700W
Decision Frequency	30 mins
Number of Time Slots	28
Initial SoC	40-50%
Target Final SoC	100%
EV Battery Capacity	24kWh
Number of EVs Simulated	10, 30, 60
Number of Days Simulated	28
Arrival/Departure Time	18:00/07:30
Peak Electricity Price	€0.2062
Off-Peak Electricity Price	€0.1019
Peak/Off-Peak Hours	09:00-00:00/00:00-09:00

Table 5.1: Summary of Experimental Parameters.

Parameter	Value
Chromosome Width	Number of time slots
Chromosome Height	Number of EVs
Population Size	100
Number of Generations	200
Crossover Rate	0.5
Mutation Rate	0.001

Table 5.2: Summary of Genetic Algorithm Parameters.

As discussed previously, an edited version of GridLAB-D is used for the simulation of the electric grid, with the alteration allowing for the control of EV charging throughout the simulation; this is based on the schedules generated by the three fitness functions used. Alongside these simulations, results are also collected from an unedited version of GridLAB-D running a simulation of a normal grid in which the cars charge as soon as they are plugged in until completion; we refer to this as the 'Greedy Approach' due to its action of taking as much power as possible until full. The expected behaviour of the greedy approach is illustrated in Figure 5.5, in which we see full load from the beginning until a sharp drop when the EVs complete charging.

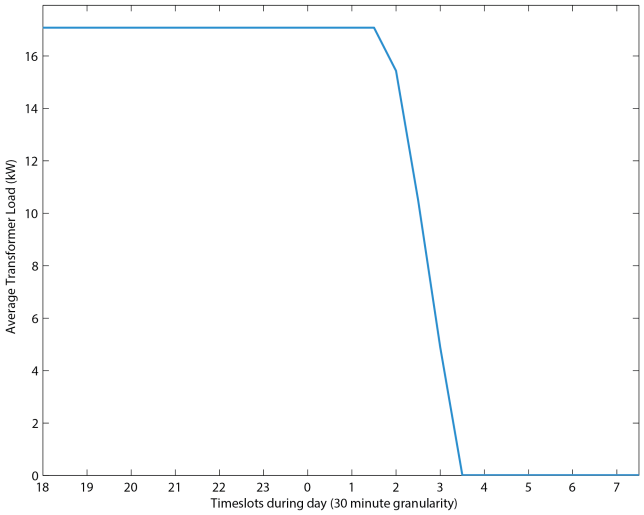

Figure 5.5: Expected load profile of the grid when charging EVs under the greedy approach.

Chapter 6

Experimental Results

This chapter will present and discuss the results obtained through the experiments undertaken. The graphs shown are generated by MATLAB from the output of GridLAB-D when utilising the schedules generated by the genetic algorithm with the three proposed fitness functions. We will focus on a number of important factors, as discussed in Chapters 2 and 3: transformer load, final state of charge (SoC), peak-to-average ratio (PAR), and the cost of electricity used when discussing these results throughout this chapter. We will also take a look toward the growth in fitness through generations within all of the fitness functions used. Throughout all of the graphs and discussion, we will be looking at the three different fitness functions used in our GAs, as well as the 'greedy approach', which is simply the basic system of all EVs charging immediately when plugged in, and continuing until reaching 100% SoC (whenever possible).

6.1 Transformer Load

In this section, we will look at the effects of the algorithms on the transformer load throughout the time periods specified. Figure 6.1 shows the transformer load within the grid at each time slot simulated for 10 EVs. This is averaged across the 28 day cycle used in the experiments, with the transformer load at each timeslot being denoted in kWh.
The plots shown in this graph represent the greedy approach and the proposed

approach with the three fitness functions detailed in Chapter 5. We can clearly see the expected behaviour from the greedy approach, in that all EVs immediately begin charging at 18:00 when they arrive home, keeping the transformers under a 17kWh load (recall that 1.7kWh is the plug-in charging value for each car specified in Table 5.1) until there is a sharp drop to 0kWh between 01:30 and 03:30 as the EVs reach 100% SoC. We can see the rate at which the load drops increase after the first timeslot; this is expected, as we should see the EVs with the highest initial SoC complete charging first, before other EVs complete charging in the following timeslots.

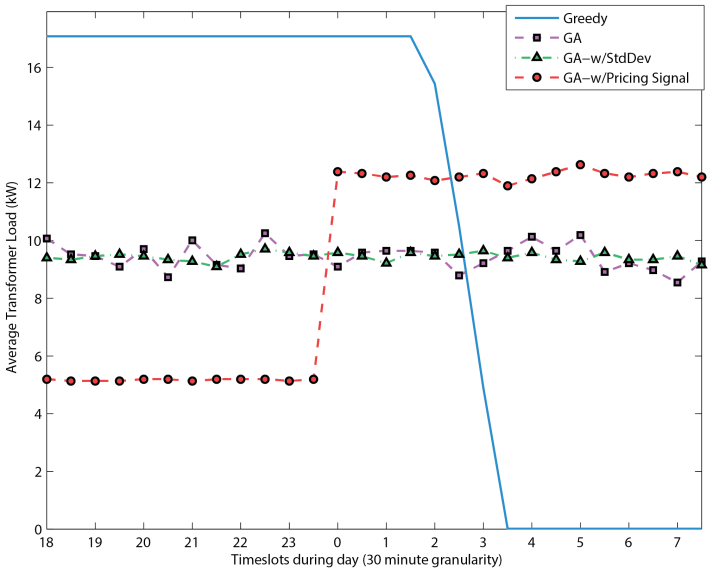

Figure 6.1: A comparison of the transformer loads under the 3 fitness functions used, as well as with the greedy approach (the results shown are for 10 EVs).

The three fitness functions show stark differences in transformer load to the greedy approach. The steady charging fitness function (denoted by GA in all figures), which simply tries to charge the EVs to 100% in a balanced way throughout the time period, shows a far lower load on the transformers, varying between 50-60% of the peak load from the greedy approach. We can see however, that it does show some fluctuations, dropping and increasing sharply throughout the time period, leading to several peaks and troughs in the transformer load.

The standard deviation fitness function (denoted by GA w/StdDev), which takes into account the standard deviation of the number of EVs charging at any one time, shows a much smoother transformer load throughout the time period, approximately at the average load of the first fitness function. The plot shows only small amounts of fluctuation in either direction, and has no real peaks or troughs in transformer load, showing a marked improvement over the first fitness function.
The pricing signal fitness function (denoted by GA w/Pricing Signal) has a very different set of goals than the first and second. This fitness function takes into account the Electric Ireland 'Nightsaver' summertime pricing signal as discussed in Chapter 5, and aims to reduce the cost of electricity to the consumers. This leads to the spike in transformer load from 23:30 to 00:00, where the price drops to almost half of the day rate. The fitness function also aims to smooth out the transformer load at either side of this spike, leading to a relatively stable transformer load from 18:00-23:30, and from 00:00-07:30.
Figures 6.2, 6.3, 6.4, and 6.5 show the same plots as 6.1 but with the standard deviation of the different loads throughout the 28-day period simulated. From these plots, we can see the stark difference in results between the steady charging function and the standard deviation function, with the latter achieving a much lower standard deviation throughout the time period despite the changes to EV initial SoC throughout this time. We can also see the zero standard deviation on either side of the load drop in the greedy approach, but with fluctuations during the drop itself; this is due to the different initial SoCs of the EVs, where a higher initial SoC EV will be switched off sooner, leading to an earlier drop than others.

Figure 6.6 shows the same plots for a simulation with 30 EVs. We can see that the greedy approach takes the same shape, but with a much higher load due to the larger number of EVs. The first fitness function also shows a very similar plot to the scenario with 10 EVs; the plot is slightly smoothed out, but still shows a number of small peaks and troughs. However, the second fitness function, which takes into account the standard deviation, shows a massive improvement with almost no fluctuation at all in the transformer load throughout the time period. We have seen from Figures 6.2, 6.3, 6.4, and 6.5 how much of an impact the

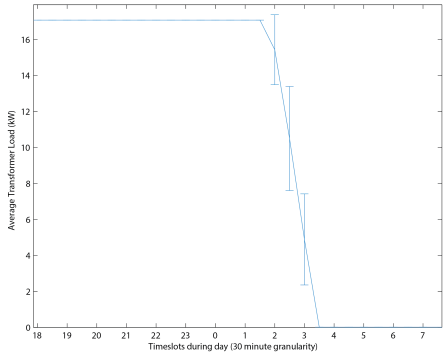

Figure 6.2: Transformer load under the greedy approach for 10 EVs; also displaying the standard deviation across the 28-day period as error bars.

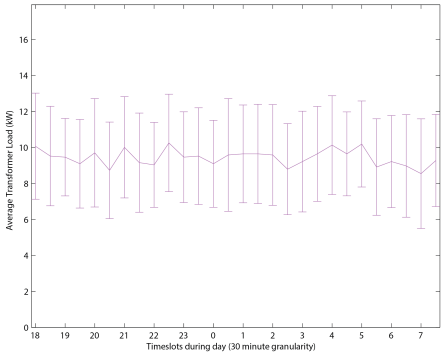

Figure 6.3: Transformer load under the GA approach with the steady charging fitness function for 10 EVs; also displaying the standard deviation across the 28-day period as error bars.

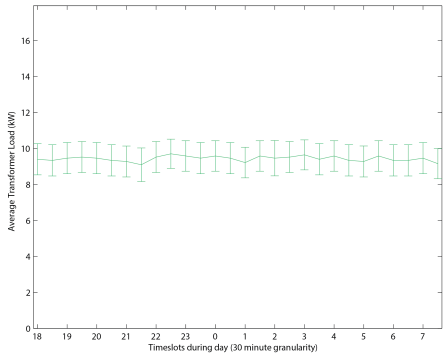

Figure 6.4: Transformer load under the GA approach with the standard deviation fitness function for 10 EVs; also displaying the standard deviation across the 28-day period as error bars.

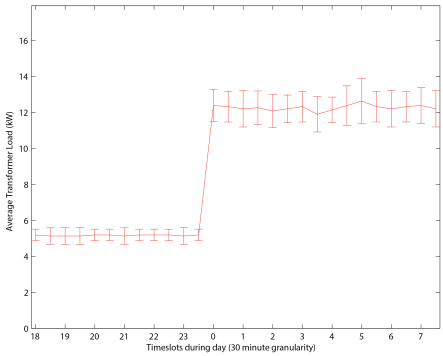

Figure 6.5: Transformer load under the GA approach with the pricing signal fitness function for 10 EVs; also displaying the standard deviation across the 28-day period as error bars.

standard deviation fitness function has on reducing the load fluctuation across the time period for 10 EVs; we will look at this closer later when discussing the PAR of the different algorithms.

The third and final fitness function shows a similar plot once again for 30 EVs, with a spike at the switchover to the night rate. We see a smaller spike however

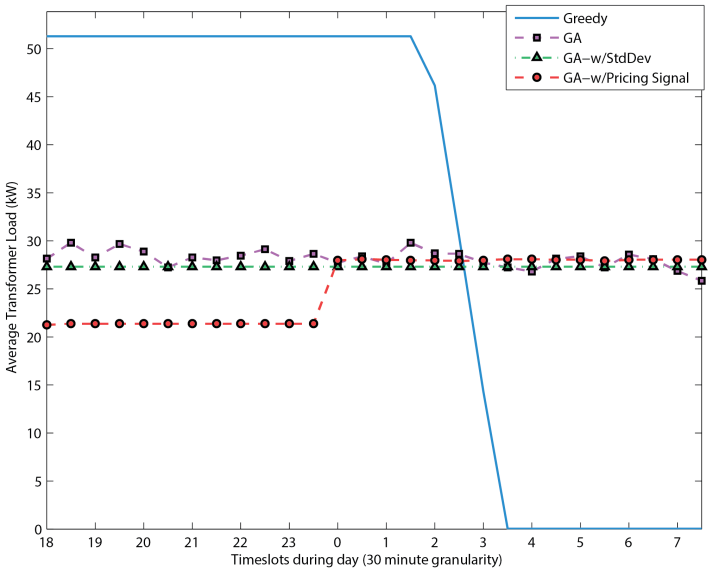

Figure 6.6: A comparison of the transformer loads under the 3 fitness functions used, as well as with the greedy approach (the results shown are for 30 EVs).

as compared with 10 EVs when using this fitness function; we will see later how this impacts the SoC and PAR.

Finally, Figure 6.7 shows the same plots for a 60 EV simulation. We can see that the greedy approach is once again following the same plot at a higher load, and the first two fitness functions operate in a similar manner when scaled to this number of EVs, with a very consistent and smooth plot once again for the second fitness function. The third fitness function however shows a smaller peak again when brought to this scale; we will examine this later alongside the results for 30 EVs.

6.2 State of Charge

In this section we look to the final SoC of the EVs when using the greedy approach, and contrast these results with the final SoC when using the three GA fitness functions developed. Figure 6.8 illustrates these results in a bar plot showing the initial SoC alongside the four separate final SoCs for the 10 EVs in

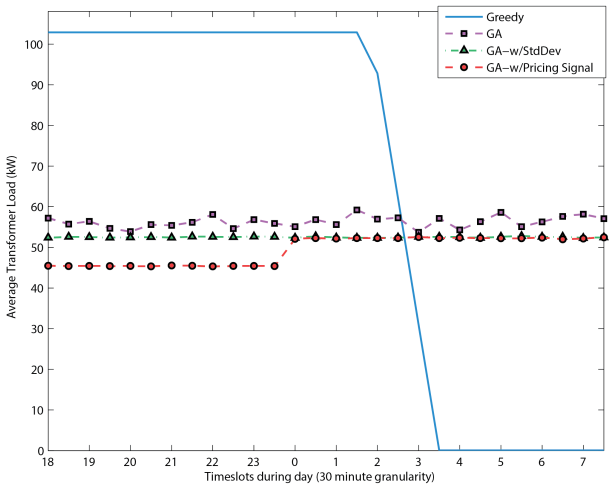

Figure 6.7: A comparison of the transformer loads under the 3 fitness functions used, as well as with the greedy approach (the results shown are for 60 EVs).

the simulation, averaged over the 28 day period.

The results show that the initial charge averages in the area of 45%, expected due to the range of 40-50% initial SoC for each EV each day. The final SoCs differ for the different algorithms.

The greedy algorithm, as expected, reaches 100% SoC for every car; this is due to the approach of charging all EVs immediately until fully charged, and only then stopping. The three forms of the GA fitness functions show final SoCs of approximately 90%, with the first two fitness functions (steady charging and standard deviation) consistently achieving above 90%, and the third fitness function (pricing signal) reaching 89-90% on average. This difference is due to how these fitness functions target different goals to just reaching 100% charge, and do not reach an absolute optimum solution within the generations under these conditions.

For the simulation with 30 EVs, illustrated in Figure 6.9, the results are almost identical; this is true in particular for the greedy approach and the first two fitness functions. The main difference with this scenario lies in the third fitness function, which now only achieves approximately 85-88% on average. This outcome is predictable through Figure 6.6, where we can see that the transformer load does not spike as high after the price shift as it does for the 10 EV scenario,

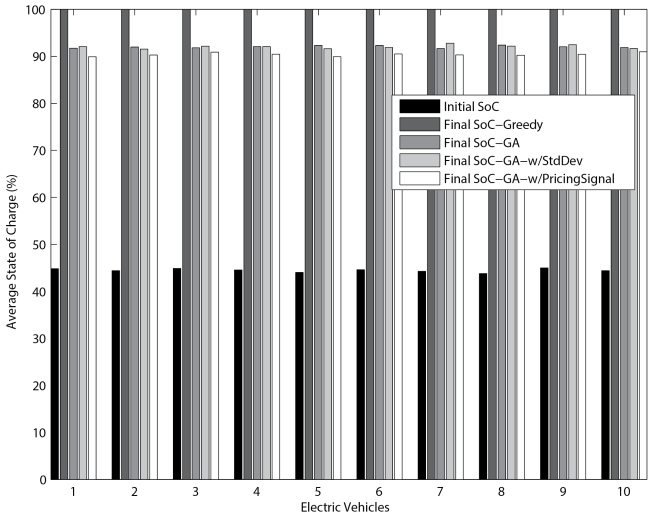

Figure 6.8: A comparison of the initial and final SoCs of the EVs under the 3 fitness functions used, as well as with the greedy approach (the results shown are for 10 EVs).

leading to a lower amount of power supplied to the EVs on average. While the SoC is not as high as with the other approaches, this fitness function has the goals of reducing price and balancing load on either side of the price change; as such, we will examine the different costs and PARs of the approaches later, so as to analyse this trade-off.

Figure 6.10 illustrates the SoC in the same way as previous plots, for a 60 EV scenario. We see once again a consistent 100% final SoC for the greedy approach, as well as the above 90% final SoC from the steady charging fitness function we have seen for 10 and 30 EV simulations. We do however see a slight drop in final SoC for the second fitness function (standard deviation) to just below 90%. This shows a slight drop in the performance of this fitness function when brought to such scale, but still within a small range. The third fitness function also shows a drop over the 30 EV simulation, but with a smaller drop than was seen from the 10 EV to the 30 EV simulation. We now see approximately 84-87% final SoC for EVs on average.

Figure 6.11 shows the initial and final SoCs at each day within the simulated

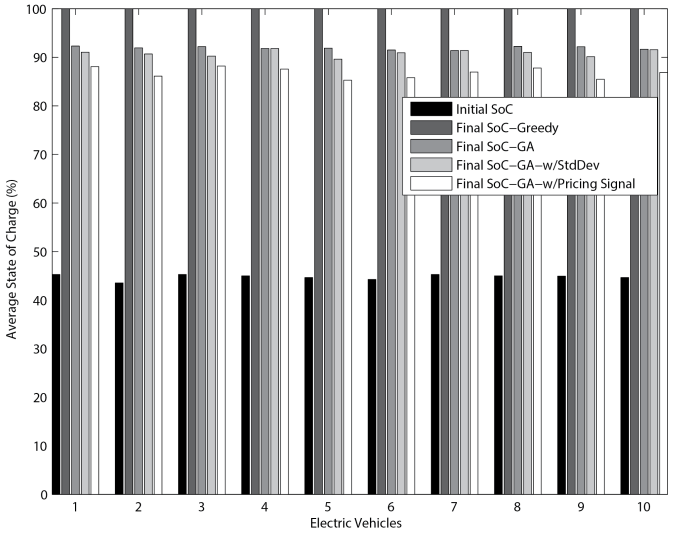

Figure 6.9: A comparison of the initial and final SoCs of the EVs under the 3 fitness functions used, as well as with the greedy approach (the results shown are for a 30 EV simulation, but show 10 EVs for clarity).

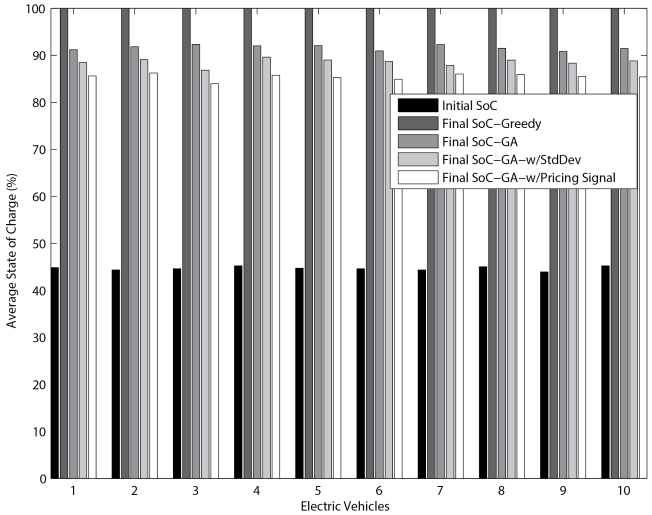

Figure 6.10: A comparison of the initial and final SoCs of the EVs under the 3 fitness functions used, as well as with the greedy approach (the results shown are for a 60 EV simulation, but show 10 EVs for clarity).

time period for a single EV picked randomly from the 10 EV simulated scenario. As we can see from this figure, the greedy approach holds a consistent 100% final SoC, while the GAs used fluctuate on either side of 90%, with two instances in which the second fitness function fell just short of 85%.

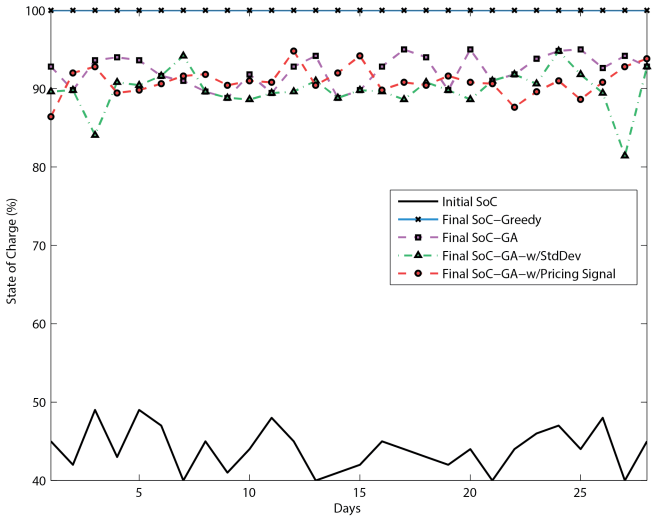

Figure 6.11: A comparison of the initial and final SoCs of a single EV under the 3 fitness functions used, as well as with the greedy approach (the results shown are for a 10 EV simulation).

Figures 6.12 and 6.13 show the same plot for single vehicles in the 30 and 60 EV simulations, showing a much more sporadic trend in final SoC for both the second and third fitness functions as compared with the 10 EV simulation in Figure 6.11.
This shows that as these functions are targeting goals other than a specific final SoC, the final SoC for individual EVs can become compromised to a degree in order to achieve better average transformer load, PAR, and electricity cost for the overall simulation.

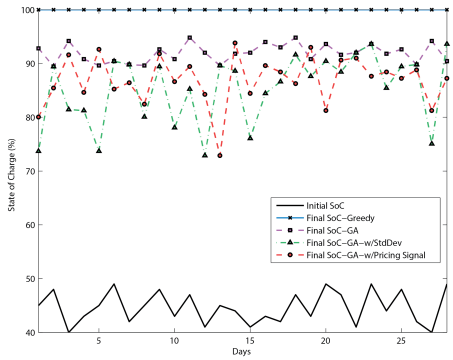

Figure 6.12: A comparison of the initial and final SoCs of a single EV in the 30 EV simulation using each approach.

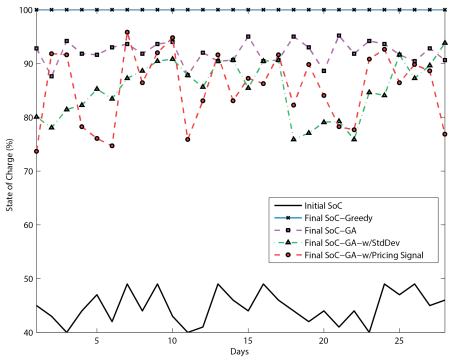

Figure 6.13: A comparison of the initial and final SoCs of a single EV in the 60 EV simulation using each approach.

6.3 Peak-to-Average Ratio

PAR is a measure of the fluctuation of the power load on the transformers in this grid simulation. It is calculated as the peak power reached divided by the average power throughout the time period. An ideal situation in this regard would be a PAR of 1.0, which would show no fluctuation from the average to the highest peak throughout the time period. Figures 6.14 and 6.15 show comparisons of the PAR for the three fitness functions within the 10 EV simulation, with 6.14 also showing the PAR from the greedy approach.

We can see from these figures that the greedy approach, as well as the first fitness function, show high PAR values throughout the simulation's 28 day time period. While the steady charging fitness function showed improvements in the overall load compared to the greedy approach, there is no such improvement seen when inspecting the PAR. We can see that the two approaches intermittently switch between being higher and lower than each other, with a number of spikes from the steady charging fitness function on days 14, 15, and 19.

Figure 6.15 shows us a clearer picture of the three fitness functions used. While the first fitness function showed no improvement in PAR to the greedy approach, we see a dramatic improvement in the second fitness function. With the added goal of a low standard deviation of transformer load, this fitness function has reduced the PAR heavily to close to the ideal figure of 1.0, ranging from approxi-

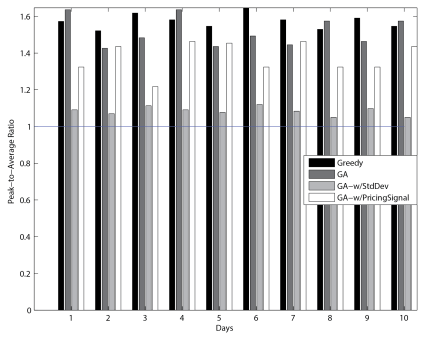

Figure 6.14: PAR comparison for the 10 EV simulation, showing the greedy approach alongside the three fitness functions used (only the first 10 days are shown for clarity).

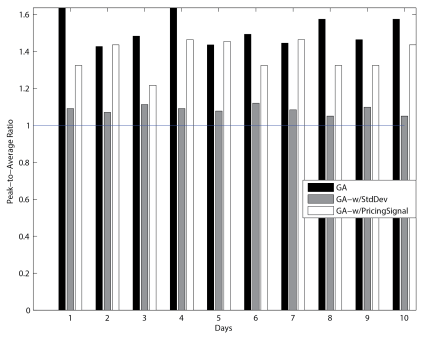

Figure 6.15: PAR comparison for the 10 EV simulation, showing only the three fitness functions used (only the first 10 days are shown for clarity).

mately 1.05 to 1.15. The third fitness function, due to its target of a low standard deviation on either side of the price drop, has a lower overall PAR than the first. It fluctuates between 1.2 and 1.45 whereas the first varies between 1.4 and 1.8, giving a much lower PAR overall despite a small number of days with a slightly higher PAR.

To analyse how the fitness functions are able to scale, we also look to the PAR in the 30 EV, and 60 EV simulations. Figures 6.16 and 6.17 show the PAR from all approaches for both of these simulations. An interesting point to note is that while the PAR of the second fitness function is generally very close to optimal, both in the 10 EV and 60 EV simulations, in the 30 EV simulation we see a consistent PAR of almost exactly 1.0.

We also see a trend of decreasing PAR for all of GAs when used with larger numbers of EVs, due to the added flexibility of the transformer load with a larger number of potential states at each time slot. This is highlighted by the fact that while the first fitness function showed a higher PAR than the greedy approach on many days within the 10 EV simulation, the algorithm shows consistently lower PAR than the greedy approach when used with 30 EVs, and the gap widens further when the simulation scales up to 60 EVs.

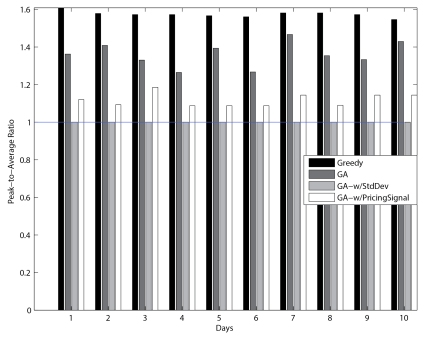

Figure 6.16: PAR comparison for the 30 EV simulation, showing the greedy approach alongside the three fitness functions used (only the first 10 days are shown for clarity).

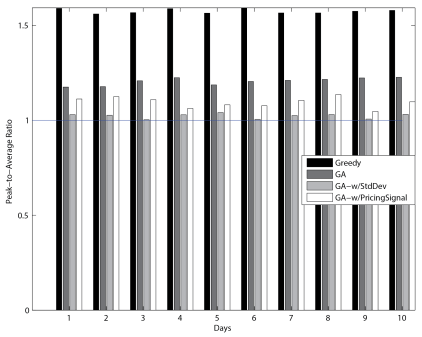

Figure 6.17: PAR comparison for the 60 EV simulation, showing the greedy approach alongside the three fitness functions used (only the first 10 days are shown for clarity).

6.4 Cost of Electricity

The cost of electricity used was the final focus of these experiments. The pricing signal fitness function specifically targets a lower overall cost to the consumer for the electricity used in charging the EVs; Figures 6.18 and 6.19 illustrate these costs for each of the approaches for the 10 EV simulation. Figure 6.18 shows the direct costs of electricity of each algorithm in the simulation. However, this representation is flawed due to the slightly lower SoC attained by the GAs when compared to the greedy approach; as such, Figure 6.19 also shows the costs of each algorithm, but with the total power supplied for each approach normalised to match the lowest power supplied by a GA for each day. This gives a precise comparison of the cost of each approach when using exactly the same number of 'units' of electricity. Figure 6.20 then shows the average cost per car for these normalised results, to better illustrate how much each consumer would pay under each approach.

From these figures we can see clearly that there is a considerable decrease in the cost of electricity to consumers using any of the GAs implemented.

Table 6.1 gives a more detailed view of the results displayed for the 10 EV simulation in Figures 6.19 and 6.20. It illustrates the breakdown of exactly when electricity is used during each approach, alongside the overall cost, and the per-

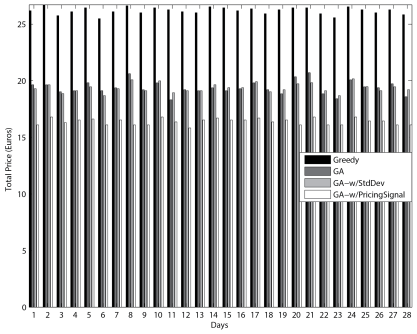

Figure 6.18: Cost comparison for the 10 EV simulation, showing the overall cost of the greedy approach alongside the three fitness functions used.

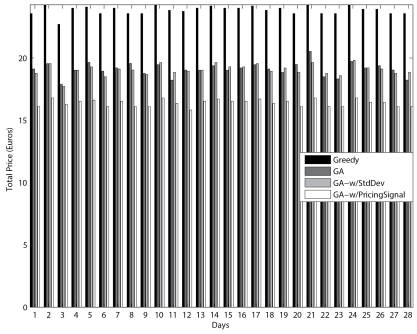

Figure 6.19: Normalised cost comparison for the 10 EV simulation, showing the overall cost of each approach, normalised to the same number of units of electricity used.

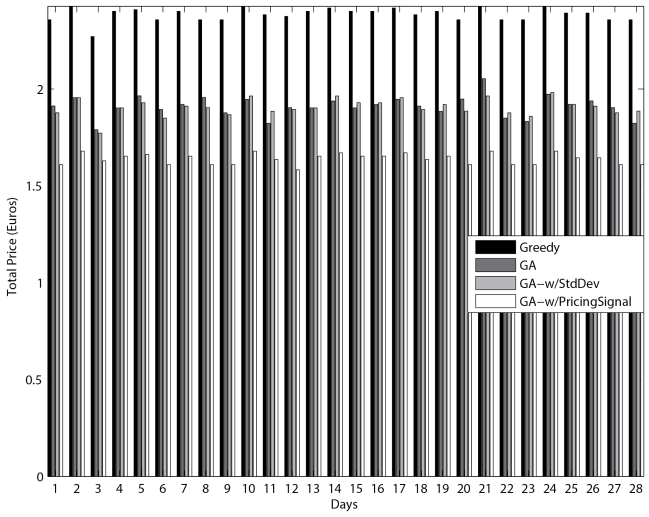

Figure 6.20: Normalised cost comparison for the 10 EV simulation, showing the average cost per car of each approach, normalised to the same number of units of electricity used.

centage savings made using each of the GAs over the greedy approach. We can see that the first two fitness functions show exactly the same cost per car, an almost 20% reduction by balancing the transformer load throughout the night.

The third fitness function however, shows an average cost reduction of almost one third (31.1%) by using the pricing signal to target a low cost. As can be seen in the usage columns, these figures use the exact same number of units overall throughout the night, so all costs are measured for the same amount of electricity supplied. Tables 6.2 and 6.3 show these results for the 30 and 60 EV simulations respectively, showing that while the first two fitness functions become even more effective at reducing costs with larger numbers of EVs, the gaps closes in the savings between the third fitness function and the steady charging/standard deviation functions.

Approach	Units (kWh) Peak	Units (kWh) Off-Peak	Cost (€) Total	Cost (€) Per Car	Cost Reduction
Greedy	102.47	26.68	23.85	2.38	-
Steady Charge	57.00	72.15	19.11	1.91	19.7%
Standard Dev.	56.58	72.57	19.06	1.91	19.7%
Pricing Signal	31.01	98.14	16.40	1.64	31.1%

Table 6.1: Analysis of Electricity Usage and Cost for the 10 EV Simulation (Averaged across 28 days).

Approach	Units (kWh) Peak	Units (kWh) Off-Peak	Cost (€) Total	Cost (€) Per Car	Cost Reduction
Greedy	307.78	44.56	68.00	2.27	-
Steady Charge	171.22	181.12	53.76	1.79	20.9%
Standard Dev.	163.80	188.53	52.99	1.77	22.0%
Pricing Signal	128.23	224.10	49.28	1.64	27.5%

Table 6.2: Analysis of Electricity Usage and Cost for the 30 EV Simulation (Averaged across 28 days).

6.5 Growth of fitness within the GAs

In order to get a full picture of the fitness functions and the results discussed, we must look toward the performance of the algorithms in a more abstract manner. Figures 6.22, 6.24 and 6.26 show the growth in the best fitness of individuals

Approach	Units (kWh) Peak	Units (kWh) Off-Peak	Cost (€) Total	Cost (€) Per Car	Cost Reduction
Greedy	617.41	73.43	134.79	2.25	-
Steady Charge	335.29	355.56	105.37	1.76	21.8%
Standard Dev.	315.26	375.59	103.28	1.72	23.4%
Pricing Signal	272.58	418.26	98.83	1.65	26.7%

Table 6.3: Analysis of Electricity Usage and Cost for the 60 EV Simulation (Averaged across 28 days).

per generation for the steady charging, standard deviation, and pricing signal fitness functions respectively. Similarly, Figures 6.21, 6.23 and 6.25 show the growth in the average fitness of the population at each generation for the three fitness functions.

We can see from the first plot (Figure 6.22) that the steady charging algorithm both achieves very close to the optimum solution quite quickly, but also scales very well to 30 and 60 EVs, still achieving almost exactly the same level of fitness with these numbers of EVs. The standard deviation function (Figure 6.24) then shows a slight decline overall due to the more complex logic implemented, but also a decrease in the fitness for the 60 EV approach, showing that the algorithm may have difficulty when the number of EVs increases further. The third plot, showing the pricing signal function (Figure 6.26), once again shows an overall decrease in fitness, again due to further complexity in the algorithm when considering the level of charge, standard deviation of load on either side of price spike, and the pricing schedule information. However, we see very little change in fitness growth between 30 and 60 EVs, which shows promise for higher numbers of EVs, particularly with the reductions in cost for the results found in the previous section. When looking at the average fitness plots (6.21, 6.23 and 6.25), we can see a similar trend to the best fitness plots, but at a slightly lower level, as would be expected.

We can see that these figures initially grow very rapidly during the early generations, but following the region of the 40th-60th generation, begin a very slow growth toward the optimum solution (1.0). We can also see that the first two fitness functions (steady charging and standard deviation) reach very close to the optimum solution, achieving almost 0.99 for the steady charging function, and

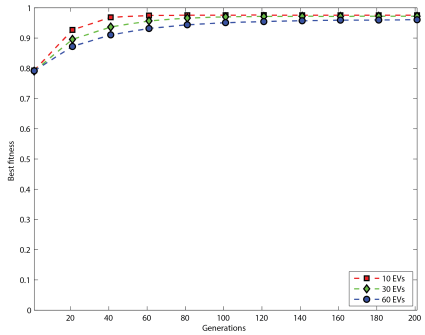

Figure 6.21: A comparison of the convergence of the steady charging fitness function under 10, 30, and 60 EV loads. The plot shows the growth in average fitness of each generation.

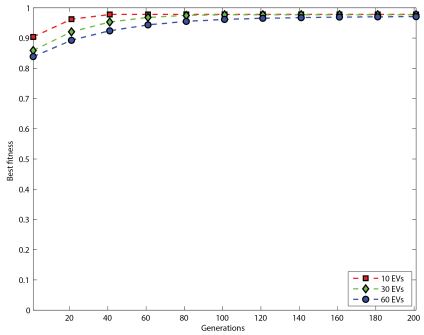

Figure 6.22: A comparison of the convergence of the steady charging fitness function under 10, 30, and 60 EV loads. The plot shows the growth in fitness of the fittest individual of each generation.

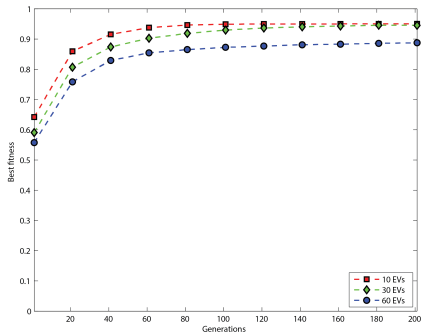

Figure 6.23: A comparison of the convergence of the standard deviation fitness function under 10, 30, and 60 EV loads. The plot shows the growth in average fitness of each generation.

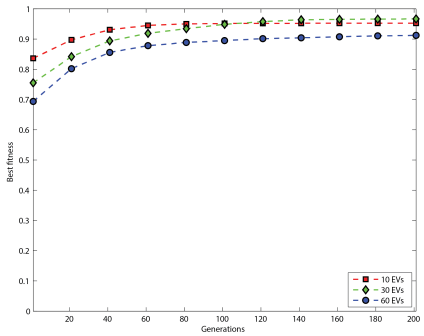

Figure 6.24: A comparison of the convergence of the standard deviation fitness function under 10, 30, and 60 EV loads. The plot shows the growth in fitness of the fittest individual of each generation.

consistently above 0.9 for the standard deviation function. The pricing signal ranges between 0.82 to 0.88 in general, showing a lower fitness than the other functions, but still generating excellent results.

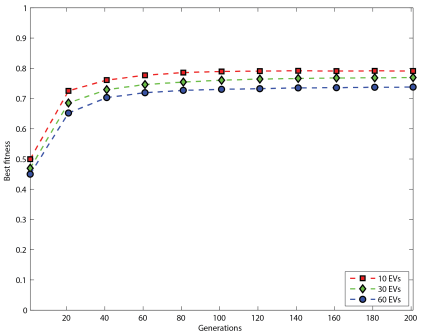

Figure 6.25: A comparison of the convergence of the pricing signal fitness function under 10, 30, and 60 EV loads. The plot shows the growth in average fitness of each generation.

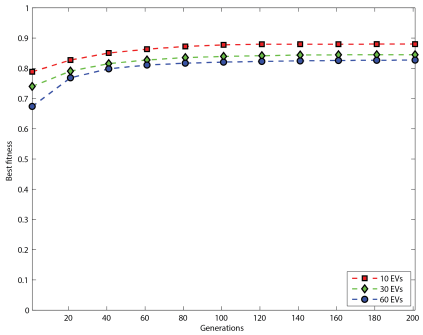

Figure 6.26: A comparison of the convergence of the pricing signal fitness function under 10, 30, and 60 EV loads. The plot shows the growth in fitness of the fittest individual of each generation.

6.6 Summary

In summary, we have seen improvements in performance from the three fitness functions under the GA approach when compared with the greedy approach. These improvements occur in PAR, average load, and cost; however, within these simulations the greedy approach does consistently achieve 100% final SoC in all scenarios, whereas the GA approach ranges from $\tilde{8}0\%$ to 95%, depending on the targets of the fitness function, and the number of EVs on the grid. This means that there is a trade-off between the two approaches in this regard; this will be discussed further in Chapter 7.

Chapter 7

Conclusions/Future Work

7.1 Conclusions from the Experimental Results

From the results discussed in the previous chapter, we can see that all of fitness functions used in our Genetic Algorithm (GA) show clear improvements in many ways over the greedy approach. We will now discuss the individual results of each fitness function.

7.1.1 Steady Charging Fitness Function

While this function showed a number of spikes in peak-to-average ratio (PAR) throughout the simulated time period, in general it showed an overall reduced PAR when compared with the greedy approach. It also showed a lower overall load on the transformers, which is a factor that needs to be taken into account when analysing PAR values. These two results combined show an reduction on the strain that the transformers in the grid would be under, potentially leading to a longer lifespan for the transformers and/or a reduction in the amount of maintenance they would require.
With relation to final SoC, we see a lower level on average to the greedy approach, but consistently within less than 10% difference. This may be undesirable for a number of reasons, some users may need 100% charge every day, or there may be adverse effects on battery lifespan when consistently only reaching 90% charge. As such, if choosing to implement this approach, or any of the other GAs

developed, this reduced expected final SoC would need to be weighed against the benefits of the greedy approach in a real-world scenario. Alternatively, time could be taken to tune these fitness functions so as to increase the final SoC, potentially at the cost of some of the benefits of PAR and transformer load.

7.1.2 Standard Deviation Fitness Function

This algorithm showed much further improvement in PAR than even the steady charging fitness function, under a 30 EV load on the grid, the second fitness function was able to consistently achieve a PAR of almost exactly 1.0. This is a huge improvement over the greedy approach and with an average transformer load at the same/slightly lower level to the first fitness function, this approach could give even better improvement in transformer lifespan and maintenance costs. These savings could be passed to the consumers in a competitive electricity market.
This approach however shows a slightly lower final SoC to the first fitness function under 30 and 60 EV loads. Once again, the fitness function could be tuned to increase the final SoC at the cost of a slightly higher average load or a slightly higher PAR. An alternate concept could be to allow users to demand 100% when needed (or at regular intervals to reduce the risk of impaired battery lifespan), at a higher cost for the electricity used, leaving other users who may need a very low final SoC the ability to avail of a lower cost to charge their EVs.

7.1.3 Pricing Signal Fitness Function

The final fitness function implemented had a different objective to the first two, this targeted a lower cost to the consumers for electricity used during the simulated time period. As is illustrated by the results in the previous chapter, this goal was achieved in all simulated scenarios. In the 10 EV simulation this fitness function reduced the costs of electricity by over a third when compared with the greedy approach, whilst the other fitness functions reduced the costs by just over 20%. This result was gained at the cost of final SoC; however, from the results in the previous chapter, we have seen that when the results are normalised to the same amount of power supplied to the EVs, there is still a reduction in cost of

over 30% over the greedy approach, as well as a reduction of just under 20% for the first two fitness functions.
The final SoC within this approach is lower than the greedy approach, as well as the two other fitness functions implemented. As with the other fitness functions implemented, this could be improved at the cost of PAR, cost, and/or average transformer load through tuning the algorithm. The use of a standard deviation calculation on either side of the price change gives us a very low overall PAR with this fitness function, so this could be sacrificed to improve the gains in other areas if deemed less important.

7.1.4 Greedy Approach

While the implemented fitness functions under the GA approach all provide improvements over the greedy approach in a number of ways, it is the only one that can consistently achieve 100% final SoC under these experimental parameters. There are many situations in which this could be the most important factor, particularly to the consumers; as such, this must be balanced against the improvements in other areas. It must also be taken into account that this, being the standard approach, requires no computation from the grid or any smart grid hardware/software to be implemented, while all other approaches would require initial investment to be implemented.

7.2 Future Work

There are a number of areas that could be investigated further through future work; in this section we will discuss some of these possibilities at a high level.

7.2.1 Real-Valued Encoding

One of the key areas that would be recommended after this dissertation would be in the use of real values in the individuals so that the EV chargers have any number of potential states of 'charging' instead of a binary on/off representation. This would mean that even though an EV charger would have a plug-in charging rate of 1700W, for example, we could generate a schedule such that in any time

slot, the EV could be charged at any power from 0W-1700W, so that a more fine-grained schedule can be generated. This could potentially lead to a further improved PAR, final SoC closer to 100%, and reduced cost to the consumer due to the higher level of control the GA would have over the schedules. This could be implemented through the use of a Real-Value encoded GA.
In the early stages of this dissertation, this concept was investigated; a basic real-valued system was implemented and designed to operate in the same way as the GAs used. However, the fitness functions used for the GAs did not transfer directly to the new real-valued encoding and did not converge on a solution in the same way as the binary approach. This led to poor performance of the results output by the GA with this encoding, despite the potential for a better result than the binary encoding; as such, these results were not included in this dissertation. The reason for the failure of the real-valued GA implemented was in the fitness functions used; it was very difficult to design a fitness function that operated well on both the real-value and binary encodings, but a fitness function designed specifically for a real-valued encoding in this scenario holds the potential to yield even better results than the approach used, and is a recommended area for future research.

7.2.2 Use of Real-World Models within GridLAB-D

The focus of this dissertation was on the charging of EVs in the grid; as such, only EVs were simulated on the grid within GridLAB-D. However, a more accurate and informed view of the grid could be used alongside this or similar logic, to show the gains of the different algorithms under these realistic circumstances. There would then also be other issues to consider, such as the peak times of basic household appliance electricity usage, and how this could be balanced alongside EV charging. Research could also be undertaken to analyse how the existing DSM ideas for household appliances could be integrated with the approach of this dissertation, to give further improvements for both electricity consumers and producers.

7.2.3 Higher Numbers of EVs & The Use of Energy Storage Units

Another area available to investigate lies in using higher numbers of EVs in the grid simulations, to the point that the standard electric grid is unable to handle to the same level as with 10, 30, or 60 EVs. Another angle would be to use a variety of battery capacities, including ones much higher than the Nissan Leaf's standard 24kWh (examples include the Tesla Roadster's 53kWh capacity, or the Tesla Model S which has 60kWh or 85 kWh capacity, depending on specification). In implementing these alternate scenarios, energy storage units could be investigated as a means of mitigating the burden on the grid.

Alternatively, in the mentioned example of a higher number of EVs, the EVs could, for example, have a higher range of initial SoC from 40%-90%, with a target of 80% to be achieved by the morning. This would mean that at peak times while appliances are used, we could use some of the >80% SoC EVs to power household appliances to alleviate the load on the grid, whilst being able to charge a small number of EVs (ones with the lowest initial SoC) simultaneously. This could potentially lead to a more balanced load on transformers, as well as proving feasibility of very large numbers of EVs on the grid.

References

[1] Great River Energy- Peaking Plants. http://www.greatriverenergy.com/makingelectricity/naturalgasoil/, . Accessed: 2014-08-27.

[2] Wisconsin Public Service- Peaking Plants. http://www.wisconsinpublicservice.com/company/peaking.aspx, . Accessed: 2014-08-27.

[3] Oglethorpe Power- Peaking Plants. http://www.opc.com/PoweringGeorgia/TypesofPowerPlants/PeakingPlants/index.htm, . Accessed: 2014-08-27.

[4] Richborough Energy Park- Peaking Plants. http://www.richboroughenergypark.co.uk/peaking.html, . Accessed: 2014-08-27.

[5] A. Mohsenian-Rad, V.W.S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *Smart Grid, IEEE Transactions on*, 1(3):320 –331, dec. 2010. ISSN 1949-3053. doi: 10.1109/TSG.2010.2089069.

[6] SMART GRID - US DoE. http://energy.gov/oe/services/technology-development/smart-grid. Accessed: 2014-08-27.

[7] A Ipakchi and F. Albuyeh. Grid of the future. *Power and Energy Magazine, IEEE*, 7(2):52–62, March 2009. ISSN 1540-7977. doi: 10.1109/MPE.2008.931384.

[8] A.J. Collin, J.L. Acosta, I. Hernando-Gil, and S.Z. Djokic. An 11 kv steady state residential aggregate load model. part 2: Microgeneration and demand-side management. In *PowerTech, 2011 IEEE Trondheim*, pages 1–8, 2011. doi: 10.1109/PTC.2011.6019384.

[9] A.J. Conejo, J.M. Morales, and L. Baringo. Real-time demand response model. *Smart Grid, IEEE Transactions on*, 1(3):236 –242, dec. 2010. ISSN 1949-3053. doi: 10.1109/TSG.2010.2078843.

[10] Edgar Galvan, Colin Harris, Ivana Dusparic, Siobhán Clarke, and Vinny Cahill. Reducing electricity costs in a dynamic pricing environment. In *Proc. Third IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 169 – 174, Tainan, Taiwan, november 2012. IEEE Press.

[11] Edgar Galván-López, Adam Taylor, Siobhán Clarke, and Vinny Cahill. Design of an automatic demand-side management system based on evolutionary algorithms. In *Proceedings of the 29th Annual Symposium on Applied Computing, SAC '14*, pages 525 – 530, Gyeongju, Korea, 26-28 March 2014. ACM.

[12] Edgar Galván-López, Colin Harris, Leonardo Trujillo, Katya Rodríguez Vázquez, Siobhán Clarke, and Vinny Cahill. Autonomous demand-side management system based on monte carlo tree search. In *IEEE International Energy Conference (EnergyCon)*, pages 1325 – 1332. IEEE Press, 2014.

[13] C.W. Gellings. The concept of demand-side management for electric utilities. *Proceedings of the IEEE*, 73(10):1468–1470, 1985. ISSN 0018-9219. doi: 10.1109/PROC.1985.13318.

[14] C.W. Gellings and William M. Smith. Integrating demand-side management into utility planning. *Proceedings of the IEEE*, 77(6):908–918, 1989. ISSN 0018-9219. doi: 10.1109/5.29331.

[15] Gilbert M. Masters. *Renewable and Efficient Electric Power Systems*. Wiley-Interscience, 2004.

[16] P. Palensky and D. Dietrich. Demand side management: Demand response, intelligent energy systems, and smart loads. *Industrial Informatics, IEEE Transactions on*, 7(3):381–388, Aug. ISSN 1551-3203. doi: 10.1109/TII.2011.2158841.

[17] Goran Strbac. Demand side management: Benefits and challenges. *Energy Policy*, 36(12):4419 – 4426, 2008. ISSN 0301-4215. doi: http://dx.doi.org/10.1016/j.enpol.2008.09.030. URL `http://www.sciencedirect.com/science/article/pii/S0301421508004606`. Foresight Sustainable Energy Management and the Built Environment Project.

[18] Hung Khanh Nguyen, J.B. Song, and Zhu Han. Demand side management to reduce peak-to-average ratio using game theory in smart grid. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 91–96, March 2012. doi: 10.1109/INFCOMW.2012.6193526.

[19] J.C. Fuller, S.E. McHann, and W. Sunderman. Using open source modeling tools to enhance engineering analysis. In *Rural Electric Power Conference (REPC), 2014 IEEE*, pages C4–1–C4–5, May 2014. doi: 10.1109/REPCon.2014.6842209.

[20] S. Ghaemi and S. Schneider. Potential analysis of residential demand response using gridlab-d. In *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, pages 8039–8045, Nov 2013. doi: 10.1109/IECON.2013.6700477.

[21] Edgar Galván-López, Ruohua Li, Constantinos Patsakis, Siobhan Clarke, and Vinny Cahill. Heuristic-based multi-agent monte carlo tree search. In *Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference on*, pages 177–182, July 2014. doi: 10.1109/IISA.2014.6878747.

[22] Edgar Galván-López, Riccardo Poli, and CarlosA.Coello Coello. Reusing code in genetic programming. In Maarten Keijzer, Una-May OReilly, Simon Lucas, Ernesto Costa, and Terence Soule, editors, *Genetic Programming*, volume 3003 of *Lecture Notes in Computer Science*, pages

359–368. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-21346-8. doi: 10.1007/978-3-540-24650-3_34. URL http://dx.doi.org/10.1007/978-3-540-24650-3_34.

[23] Edgar Galván-López. Efficient graph-based genetic programming representation with multiple outputs. *International Journal of Automation and Computing*, 5(1):81–89, 2008. ISSN 1476-8186. doi: 10.1007/s11633-008-0081-4. URL http://dx.doi.org/10.1007/s11633-008-0081-4.

[24] Edgar Galván-López, JohnMark Swafford, Michael ONeill, and Anthony Brabazon. Evolving a ms. pacman controller using grammatical evolution. In Cecilia Di Chio, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Anik Ekrt, AnnaI. Esparcia-Alcazar, Chi-Keong Goh, JuanJ. Merelo, Ferrante Neri, Mike Preu, Julian Togelius, and GeorgiosN. Yannakakis, editors, *Applications of Evolutionary Computation*, volume 6024 of *Lecture Notes in Computer Science*, pages 161–170. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12238-5. doi: 10.1007/978-3-642-12239-2_17. URL http://dx.doi.org/10.1007/978-3-642-12239-2_17.

[25] Edgar Galván-López, James McDermott, Michael O'Neill, and Anthony Brabazon. Towards an understanding of locality in genetic programming. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 901–908, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0072-8. doi: 10.1145/1830483.1830646. URL http://doi.acm.org/10.1145/1830483.1830646.

[26] Edgar Galván-López and Riccardo Poli. Some steps towards understanding how neutrality affects evolutionary search. In ThomasPhilip Runarsson, Hans-Georg Beyer, Edmund Burke, JuanJ. Merelo-Guervs, L.Darrell Whitley, and Xin Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 778–787. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-38990-3. doi: 10.1007/11844297_79. URL http://dx.doi.org/10.1007/11844297_79.

[27] Edgar Galván-López, Riccardo Poli, Ahmed Kattan, Michael ONeill, and Anthony Brabazon. Neutrality in evolutionary algorithms what

do we know? *Evolving Systems*, 2(3):145–163, 2011. ISSN 1868-6478. doi: 10.1007/s12530-011-9030-5. URL http://dx.doi.org/10.1007/s12530-011-9030-5.

[28] R. Poli and E. Galván-López. The effects of constant and bit-wise neutrality on problem hardness, fitness distance correlation and phenotypic mutation rates. *Evolutionary Computation, IEEE Transactions on*, 16(2):279–300, April 2012. ISSN 1089-778X. doi: 10.1109/TEVC.2011.2132726.

[29] Edgar Galván-López, James McDermott, Michael ONeill, and Anthony Brabazon. Defining locality as a problem difficulty measure in genetic programming. *Genetic Programming and Evolvable Machines*, 12(4):365–401, 2011. ISSN 1389-2576. doi: 10.1007/s10710-011-9136-3. URL http://dx.doi.org/10.1007/s10710-011-9136-3.

[30] Edgar Galván-López, Stephen Dignum, and Riccardo Poli. The effects of constant neutrality on performance and problem hardness in gp. In *Proceedings of the 11th European Conference on Genetic Programming*, EuroGP'08, pages 312–324, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78670-8, 978-3-540-78670-2. URL http://dl.acm.org/citation.cfm?id=1792694.1792723.

[31] James McDermott, Edgar Galván-López, and Michael ONeill. A fine-grained view of gp locality with binary decision diagrams as ant phenotypes. In Robert Schaefer, Carlos Cotta, Joanna Koodziej, and Gnter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 164–173. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15843-8. doi: 10.1007/978-3-642-15844-5_17. URL http://dx.doi.org/10.1007/978-3-642-15844-5_17.

[32] NguyenQuang Uy, NguyenXuan Hoai, Michael ONeill, R.I. McKay, and Edgar Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011. ISSN 1389-2576. doi: 10.1007/s10710-010-9121-2. URL http://dx.doi.org/10.1007/s10710-010-9121-2.

[33] E. Galván-López, B. Cody-Kenny, L. Trujillo, and A Kattan. Using semantics in the selection mechanism in genetic programming: A simple method for promoting semantic diversity. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2972–2979, June 2013. doi: 10.1109/CEC.2013.6557931.

[34] GridLAB-D Home Page. `http://www.gridlabd.org/`. Accessed: 2014-08-24.

[35] Pyevolve Introduction Page. `http://pyevolve.sourceforge.net/intro.html`. Accessed: 2014-08-24.

Appendix

Figures 1 and 2 illustrate the simulation scenario of 10 EVs using 3300W plug-in charging power.

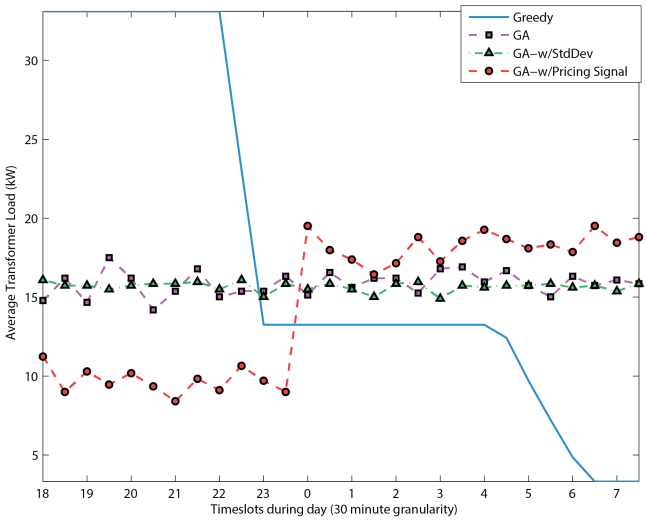

Figure 1: A comparison of the transformer loads under the 3 fitness functions used, as well as with the greedy approach (the results shown are for 10 EVs charging at 3300W).

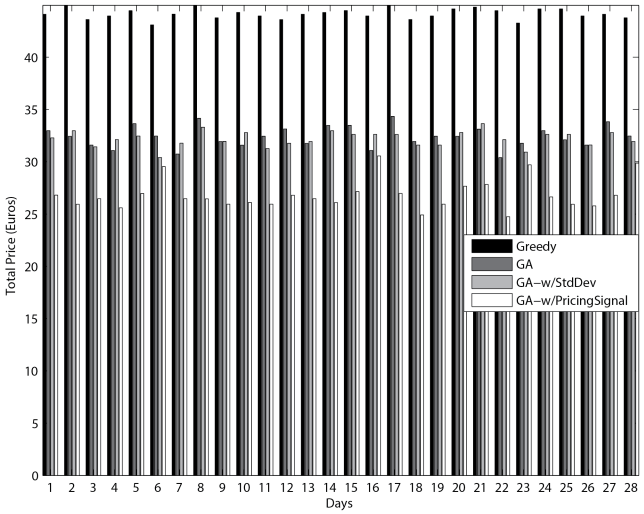

Figure 2: Cost comparison for the 10 EV simulation charging at 3300W, showing the overall cost of the greedy approach alongside the three fitness functions used.