Using Evolutionary Programming to Increase the Accuracy of an Ensemble Model For Energy Forecasting

James Gramz
Marquette University

USING EVOLUTIONARY PROGRAMMING TO INCREASE THE ACCURACY OF AN ENSEMBLE MODEL FOR ENERGY FORECASTING

by

James Gramz, B.S.

A Thesis Submitted to the Faculty of the
Graduate School, Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

May 2014

ABSTRACT

USING EVOLUTIONARY PROGRAMMING TO INCREASE THE ACCURACY OF AN ENSEMBLE MODEL FOR ENERGY FORECASTING

James Gramz, B.S.

Marquette University, 2014

Natural gas companies are always trying to increase the accuracy of their forecasts. We introduce evolutionary programming as an approach to forecast natural gas demand more accurately. The created Evolutionary Programming Engine and Evolutionary Programming Ensemble Model use the current GasDay models, along with weather and historical flow to create an overall forecast for the amount of natural gas a company will need to supply to their customers on a given day. The existing ensemble model uses the GasDay component models and then tunes their individual forecasts and combines them to create an overall forecast.

The inputs into the Evolutionary Programming Engine and Evolutionary Programming Ensemble Model were determined based on currently used inputs and domain knowledge about what variables are important for natural gas forecasting. The ensemble model design is based on if–statements that allow different equations to be used on different days to create a more accurate forecast, given the expected weather conditions.

This approach is compared to what GasDay currently uses based on a series of error metrics and comparisons on different types of weather days and during different months. Three different operating areas are evaluated, and the results show that the created Evolutionary Programming Ensemble Model is capable of creating improved forecasts compared to the existing ensemble model, as measured by Root Mean Square Error (RMSE) and Standard Error (Std Error). However, the if–statements in the ensemble models were not able to produce individually reasonable forecasts, which could potentially cause errant forecasts if a different set of if–statements are true on a given day.

ACKNOWLEDGMENTS

James Gramz, B.S.

The completion of this thesis would not have been possible if not for the guidance and encouragement of my family, colleagues, and committee members, Dr. Ronald Brown, Dr. George Corliss, and Dr. James Richie. I would specifically like to thank Dr. Corliss for the many hours spent with me offering his guidance, expertise, encouragement, and advice throughout my undergraduate and graduate career. I would also like to thank Dr. Brown and the GasDay Lab for the financial support that enabled me to fulfill my dream.

I would like to express my thanks to my colleagues and friends, Paul Kaefer, James Lubow, Nick Winninger, Tian Gao, and Hermine Akouemo, for sharing ideas and offering help and advice during my graduate studies. It was a pleasure working with all of you in pursuing a common goal.

I dedicate this work to my parents, Ray and Sharon, and my sister Sandy, for the love and support they have provided me throughout this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

THESIS INTRODUCTION

This chapter presents an introduction to the natural gas industry, why accurate gas forecasts are needed, and an explanation of the GasDay Lab where this thesis is being completed. The final sections of this chapter will explain the problem that is being addressed and the proposed solution.

This thesis will focus on the current ensemble model that is in use in the GasDay Lab and how evolutionary programming can help improve the accuracy of our natural gas demand forecasts. We will use this approach to create individual ensemble models for different operating areas for which GasDay forecasts demand, and these ensemble models will provide accurate forecasts for the amount of natural gas that will be used in the given operating area.

1.1 Gas Industry

Natural gas is a fossil fuel made up of hydrocarbon gasses that include ethane, propane, butane, pentane and methane, along with hydrogen sulfide, carbon dioxide, oxygen, and nitrogen [24]. Methane is the most desired gas and is what is assumed when talking about natural gas, unless otherwise specified. Natural gas is produced by gas and oil companies, as they are commonly found together, and by

independent gas companies. The collected gas is traded in a competitive market and transported across the United States through a series of underground pipes. Due to federal regulations, the industry is broken up into three main sections: production, transportation, and distribution. The production part includes the wellheads (both on and off shore) that collect the natural gas and put the gas into lines that go to their processing plants to remove any unwanted gases. The gas is then pushed into gas lines that are owned by transportation companies. Transportation companies make their money from Local Distribution Companies (LDC's) by receiving a fee for the amount of gas that they carry to the company. The gas is moved by pumping stations that are located every 40 – 100 miles, depending on how much natural gas is flowing through the pipe [25]. Compressor stations are used to increase the pressure in the pipes to allow the gas to continue to move. Local Distribution Companies take the gas they receive and sell it to the end users who actually burn the gas. Local Distribution Companies can also act as a transport company to a user if they bought the gas through a 3rd party vendor. A diagram of the entire route from production to use is shown in Figure 1.1.

Natural gas consumption is decomposed into five categories of end users [36]:

- **Industrial** – Industrial customers are the biggest users of natural gas by volume [12]. The gas that they use is relatively constant throughout the year, as they are making a product year round. Usually the bulk of Base Load, or

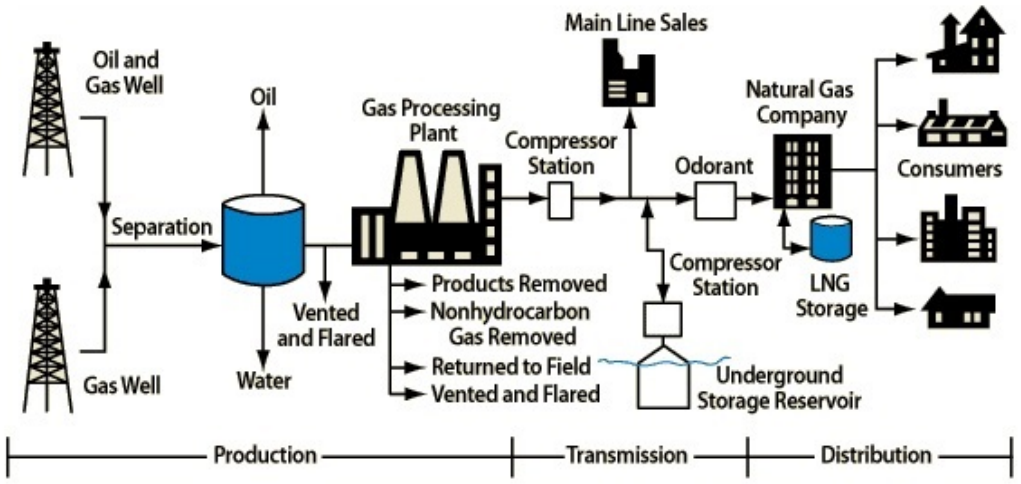

Figure 1.1: Gas distribution network from the ground to the end user [11]

non–temperature sensitive load, comes from these customers. Their use often has a weekly pattern since many production lines are shut down on weekends.

- **Commercial** – Commercial customers are also big users of natural gas and fall in between Industrial and Residential customers. They use the gas for heating, but are normally large buildings such as schools, hospitals, hotels, and government buildings. Since they use natural gas for heating, their use is referred to as Heat Load.

- **Residential** – Residential customers are by far the most common end user of natural gas in the United States by total number. Residential customers' usage of natural gas is the most dependent on the temperature. They use more gas during the winter months and less during the summer months, giving rise to a cyclical pattern year after year with peak loads typically

during January and February. Since the customers' usage is so dependent on the temperature, their use is referred to as Heat Load.

- **Power Generation** – Natural gas is very clean to burn, and it is easy to control the amount of energy it produces. The amount of natural gas used for electric power generation has been increasing rapidly over the past several decades. Unlike coal, once a burner is turned on, it can immediately start producing power. This use is something that GasDay does not take into account when forecasting the amount of gas a given Local Distribution Company is going to need to supply. This forecasting is normally done by the electrical industry, and they tell the natural gas companies how much gas they believe they are going to need each day.
- **Vehicle Fuel** – This has only started to account for a measurable amount of natural gas in the past 15–20 years, as the popularity of natural gas vehicles has been increasing. It is still only a very small amount compared to the other uses.

Figure 1.2 represents different customers and their use of natural gas since 1950 [6]. The use of natural gas has been nearly constant over the past few decades for commercial and residential customers, while its use for vehicle fuel and power generation has been climbing, as a greater push to reduce emissions has been taking place. Industrial use of natural gas had the opposite growth over the past two

decades, steadily decreasing as large users have moved outside of the United States for financial reasons.

In Figure 1.2, the amount of gas used is measured in millions of cubic feet. Another common unit for discussing gas quantities is a British Thermal Unit (BTU). A BTU is the amount of energy required to heat one pound of water by 1°F at one standard atmosphere (14.69 psi) or approximately 1,055 joules [10]. A single cubic foot of natural gas has approximately 1,030 BTU's of energy or 1,086,650 joules [44]. A decatherm (Dth), which will be used throughout this thesis, is just less than 1,000 cubic feet of natural gas. A Dth contains approximately 1,054,804,000 joules of energy [18]. This value is used because most of GasDay's customers discuss gas quantities in this unit or in Thousands of Dth's (MDth's), which is 1000 Dths.

1.2 Need for Accurate Forecasting of Natural Gas

Natural gas accounts for about 22% [36] of the energy used by the United States. As a nation, we use about 25.46 trillion cubic feet of natural gas per year [37] in residential, commercial, and industrial locations. This makes accurate estimates of the amount of gas a Local Distribution Company is going to use a crucial part of their operations. As discussed earlier, to get the gas to the end user from the point of capture, transportation companies are used. Every day, gas companies buy and sell natural gas on the open market, similar to every other commodity. If a Local

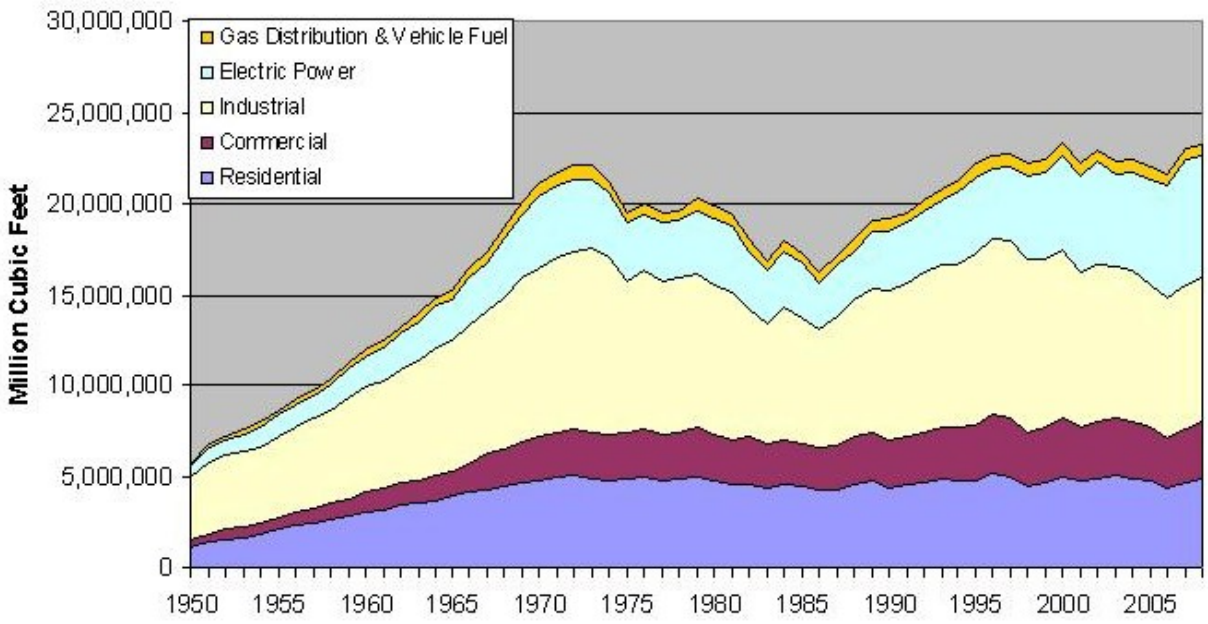

Figure 1.2: Gas usage for the five most common users of natural gas [6]

Distribution Company does not have enough natural gas to meet its demands, it must buy more, turn off certain interruptible customers, or pull some from their stored reserves. Interruptible customers are special customers that sign an agreement indicating that the Local Distribution Company may turn off their gas if they need to, in exchange for a lower charge for the gas they consume. If a Local Distribution Company has more supply than is demanded, they must sell their excess supply, put it into storage, or pay a fine to the transportation company, since they did not allow the transportation company to sell the gas to another company and make a profit.

Due to the small margin of error between what a Local Distribution Company is allowed to use based on what they originally nominated, which is set between the individual Local Distribution Companies and the transportation companies, there is a great demand for a service like GasDay to assist companies in accurately forecasting natural gas demand.

1.3 GasDay Lab

Dr. Ronald Brown, a professor in the Department of Electrical and Computer Engineering at Marquette University, founded GasDay in the summer of 1993, when Wisconsin Gas (now part of WE Energies) approached him about helping them forecast the amount of natural gas they would need for the coming day. GasDay uses weather data, actual usage, domain knowledge, and a series of mathematical models and algorithms to develop a fairly accurate forecast for how much natural gas a Local Distribution Company is going to need to supply to their customers. GasDay currently is being used at utilities in 24 states and includes over 170 different operating areas accounting for over 20% of the natural gas used by residential, commercial, and industrial customers in the United States. A graph of every state in which GasDay helps forecast natural gas demand is shown in Figure 1.3.

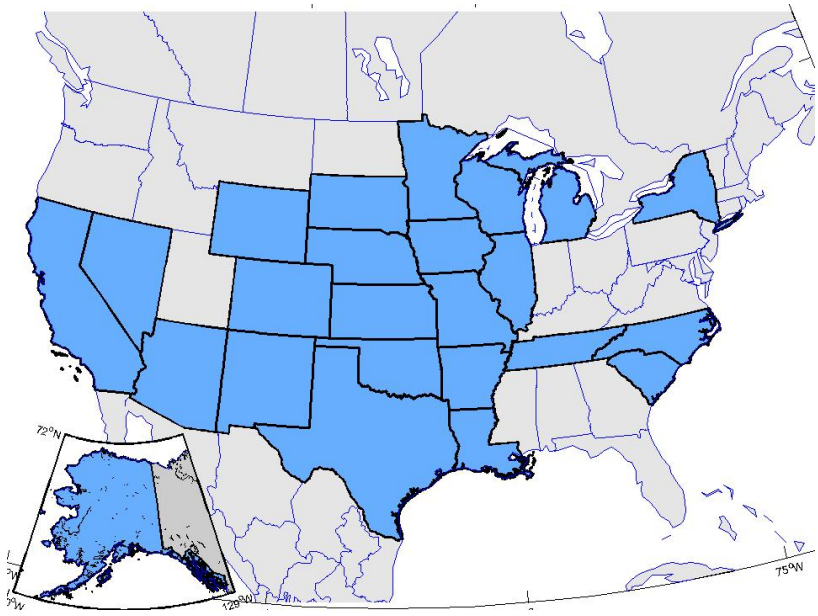

Figure 1.3: States where GasDay is used for forecasting (shaded blue)

In the past 20 years, over 200 students have been part of the GasDay project and have worked on the different mathematical models and algorithms that are used to forecast the natural gas usage of the various Local Distribution Companies.

1.4 Problem with the Ensemble Model

GasDay would like to improve the accuracy of its gas forecasts to allow Local Distribution Companies to forecast more accurately the gas that their customers will consume. The ensemble model, also known as the Dynamic Post Processor or DPP in the GasDay Lab, currently helps adjust the forecasts of the two or four different models GasDay uses to forecast natural gas demand. A diagram of this is shown in Figure 2.6, where the ensemble model is combining a Linear Regression

(LR) model and an Artificial Neural Network (ANN) model. More detail on how the GasDay Dynamic Post Processor combines two of the component model forecasts is given in Chapter 2. Instead of continuing to improve upon the approach of an ensemble model of the two or four different component models, a new ensemble model is proposed.

1.5 Proposed Solution

To address inaccuracies in the current ensemble model, a new ensemble model has been created based on the idea of a genetic algorithm. The genetic algorithm takes many different inputs based on GasDay's domain knowledge. The genetic algorithm determines the function of inputs that provides a better forecast of the actual flow than what GasDay currently has, by changing the function to find the combination that minimizes overall error. The function of inputs is expected to change depending on which data is used for training, because each part of the country should have a slightly different function, and the genetic algorithm allows for these variations. As an example, Figure 1.4 shows the percent error for a single operating area; the genetic algorithm will try to reduce this error. By increasing the accuracy of the GasDay forecasts, Local Distribution Companies will be able to forecast more accurately the amount of gas they need to have available to meet their customers' demand.

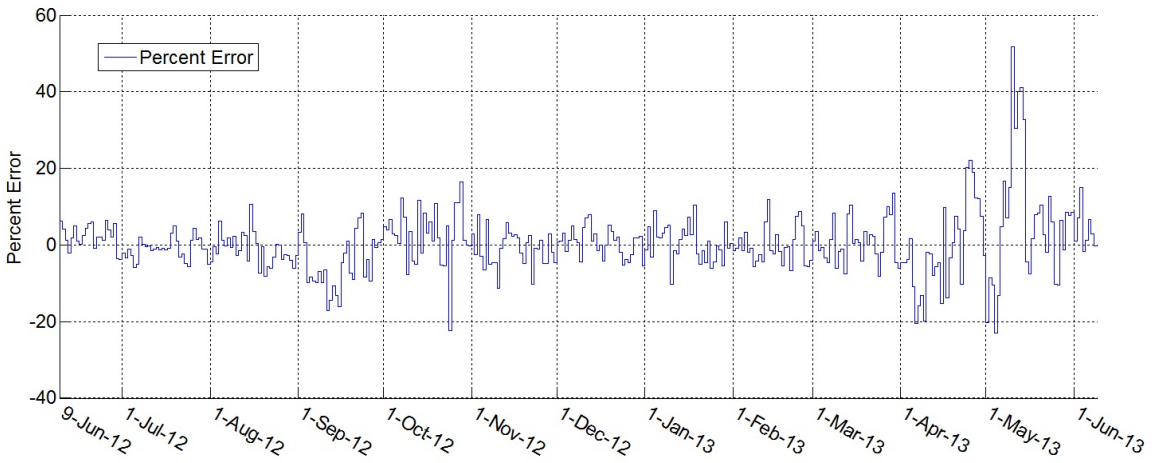

Figure 1.4: Percent error from a Local Distribution Company operating area using the current Dynamic Post Processor

1.6 Thesis Outline

This thesis consists of six chapters. The first chapter introduces the natural gas industry, customer types, need for accurate forecasts, background of the GasDay Lab, the problem for this thesis, and the proposed solution. Chapter 2 will give a detailed account of previous work completed in the area of ensemble forecasting, evolutionary programming, and describe how the GasDay process handles ensemble forecasting now. Chapter 3 offers motivations for this work and our contributions to the area, especially our process for creating new ensemble models. Chapter 4 and 5 will present the results of this work. In these chapters, all of the actual flow values have been scaled to protect the identity of the Local Distribution Company whose data is being used. The last chapter will summarize the findings and offer insights on how others can build on this work.

CHAPTER 2

CURRENT PRACTICES FOR ENSEMBLE FORECASTING

This chapter introduces current practices in ensemble forecasting, presents a detailed description of evolutionary programming, and what is currently used by GasDay for our ensemble model. This chapter also gives a description of the error metrics that will be used to evaluate the evolutionary programming approach.

2.1 Ensemble Forecasting Introduction

Ensemble forecasting is the art of forecasting with multiple forecasts, where each carries a certain weight towards the overall forecast [34, 50], as suggested in Figure 2.1. In the diagram, the overall forecast is generated by weighting each forecast. However, there are multiple ways that ensemble forecasting can be done, some of which are explained in the following sections.

2.2 Ensemble Techniques

Based on the current literature, there are multiple ways in which the current GasDay ensemble model, referred to as the Dynamic Post Processor, could be altered to improve upon the forecasts' it generates. We could add more ensemble

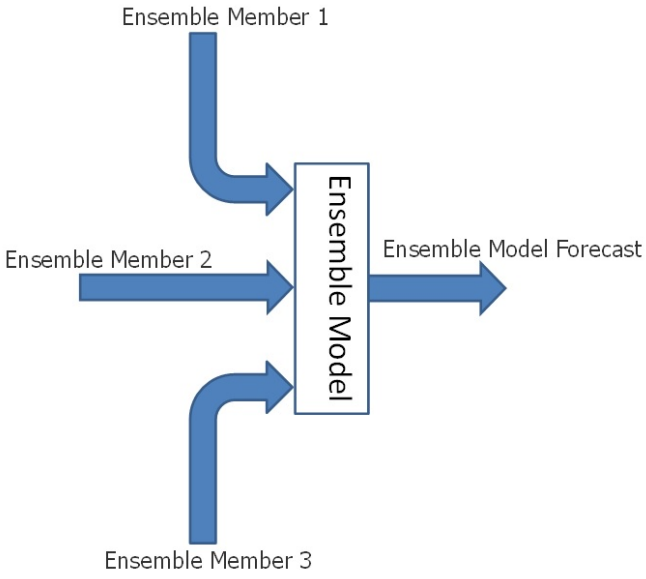

Figure 2.1: Example of an ensemble model forecast

members into the ensemble model. A similar approach was used by [26, 27, 34, 41, 47, 50] to take multiple models and combine them to increase the accuracy of the item that they were trying to forecast. A similar strategy will be implemented in this work by combining estimates from linear regression models and artificial neural network models with some weather data, past performance numbers, and past flow to produce a more accurate ensemble model forecast, as described in Chapter 3.

Another common way to improve the accuracy of an ensemble model is to improve the accuracy of the individual component models for time–series forecasting by an artificial neural network with slight variations. Wang and Wu [41] complement a neural network with a wavelet support vector machine. A wavelet

support vector machine is a specialized form of the support vector machine that was first introduced by Vapnik [39]. It can be used for pattern recognition, classification, and regression. The support vector machine maps the data from an input space to a high-dimensional feature space. The problem then becomes finding the equation to define a separating hyper-plane [42, 49]. The main differentiating feature of support vector machines is the kernel that is used. A kernel is simply a class of algorithms used for pattern recognition and analysis. The kernel can be Gaussian, polynomials, or Marr Wavelet, to name a few. The main advantages the support vector machine has over a neural network is that support vector machines automatically select the underlying network structure, have a faster convergence speed, are not prone to over-fitting, and do not get stuck in local minima [42]. As will be discussed in Section 2.6, GasDay uses an artificial neural network as one of the base component models used by the current Dynamic Post Processor. Changing the way one of the components is calculated would allow a better estimate to go into the Dynamic Post Processor and would help produce a better forecast estimate for the amount of gas that will be used in a given area.

A third common way to combine multiple forecasts is through a Bayesian approach with partial least squares. The method used by Pan and Wu [27] to help meteorologists predict rainfall used artificial neural networks with a Bayesian alteration to limit the complexity of the model. Pan and Wu assert that the

complexity of the neural network in terms of hidden layers and the multiplying constants should match the complexity of the problem that is being solved. This can be accomplished by architecture selection or regularization techniques.

Architecture selection is how the different layers of the neural network are connected and how many layers there are between the input layer and the output layer. A diagram showing two different architectures is shown in Figure 2.2. Regularization techniques encourage the network to favor smaller weights, which prevents the neural network from over-fitting to the data [27]. Figure 2.3 shows two different neural network nodes with coefficients. The node on the right is preferred over the one on the left, assuming the outputs are very close, because the coefficient values are smaller. The equation for a regularization technique is

$$\text{Total Error} = \text{Error from Output} + \sum_{i=1}^{n} \text{Coefficients}_i^2, \tag{2.1}$$

where the 2^{nd} term is the sum of the squared coefficient values. By adding in the sum of the squared coefficient values, regularization is adding another term to the cost function to help prevent the neural network from being over-trained. Other work in this area includes [9] and [47].

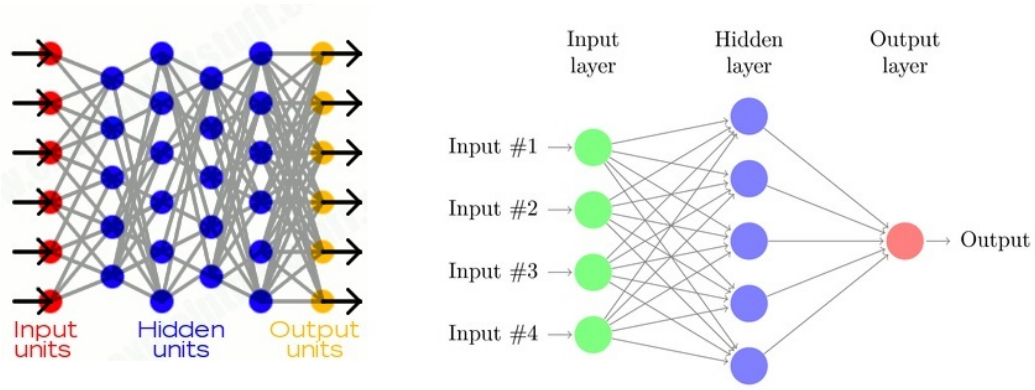

Figure 2.2: Neural network architectures [35, 45]

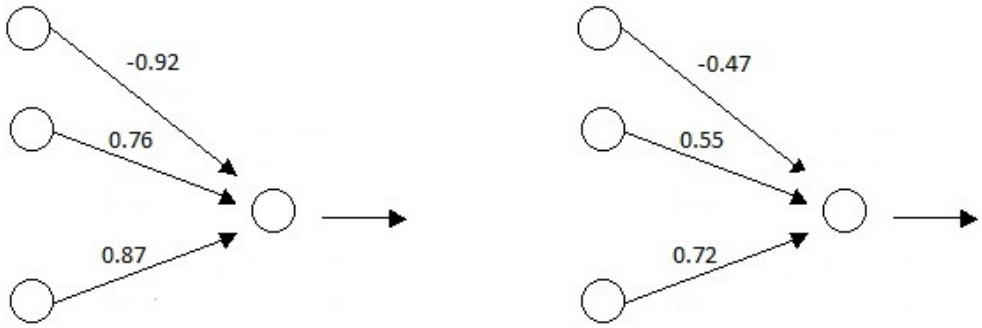

Figure 2.3: Neural network regularization techniques

Others approach time–series forecasting with an emphasis on the stationarity of the time–series data; e.g., [5, 48]. Unreliable results can be obtained if stationarity does not hold. A time–series data set is non-stationary if the mean and variance vary more than some percent as the time–series data set increases in time [5]. If the mean and variance are static, they are nearly constant throughout the entire time–series and do not follow a trend. Another time–series forecasting

technique uses ARIMA models [26, 50], which often are used to make time–series data sets stationary.

2.3 Error Modeling Techniques

We describe five of the many different error metrics:

- Mean Absolute Percent Error (MAPE),
- Mean Absolute Error (MAE),
- Mean Percent Error (MPE),
- Root Mean Square Error (RMSE), and
- Standard Error (Std Error).

Each of these metrics is preferred in different circumstances. We discuss each in turn. They are all listed here for completeness and will be used to evaluate the evolutionary programming approach presented in Chapter 3 to show how it compares to the currently implemented method. For a full description of these, as well as many others, refer to "25 Years of Time–Series Forecasting" [15].

Mean Absolute Percent Error,

$$\text{MAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{\text{actual}_t - \text{expected}_t}{\text{actual}_t} \right|, \tag{2.2}$$

which has results in percentages, is a method for determining average error over multiple data points. Mean Absolute Percent Error has a problem that if the actual value is "0," a Mean Absolute Percent Error cannot be calculated.

Mean Absolute Error,

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^{n} | \text{ actual}_t - \text{expected}_t \text{ } | \text{ }, \tag{2.3}$$

measured in units associated with the actual and expected values, is a metric used to discuss how close an estimate is to the observed outcome. This metric, similar to Mean Absolute Percent Error, is expressed as an average for all of the data points analyzed, but does not take into account the magnitude of the values. If the values are small, this metric is very susceptible to outliers that drastically increase the error.

Mean Percent Error,

$$\text{MPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{\text{actual}_t - \text{expected}_t}{\text{actual}_t}, \tag{2.4}$$

measured in percentages, is the average of the percent errors for a given data set. Mean Percent Error is ideal for determining which way the estimates are biasing, because the errors are allowed to cancel each other out since the absolute value of the errors are not taken. However, if the goal is to determine the overall accuracy of the estimates, there are better metrics to use.

Root Mean Square Error,

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^{n} (\text{actual}_t - \text{expected}_t)^2}, \tag{2.5}$$

measured in units associated with the actual and expected values, is an error metric commonly used to measure how well a model does compared to the actual value. It is a good measure of how accurate an estimated series is to the actual values, but also suffers from the same problem as Mean Absolute Error; it is dependent on the magnitude of the values for comparison.

Standard Error,

$$\text{Std Error} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}((\text{actual}_t - \text{expected}_t) - \mu)^2}\,, \tag{2.6}$$

where $\mu = \texttt{mean}(\text{actual} - \text{expected})$, is measured in units associated with the actual and expected values. Standard error is an error metric that is commonly used by GasDay to compare forecast and actual values. This value is similar to Root Mean Square Error, except the mean of the differences for all t is subtracted before the values are squared.

Although all five of the error metrics are preferred in different circumstances, if a model is better than another in one of the metrics, it is often better in every metric.

2.4 Genetic Algorithm

An introduction to genetic algorithms is helpful to understand evolutionary programming. A genetic algorithm starts with a random set of possible solutions, and through different generations, improves on the initial guesses to get to a better solution. Each of the possible solutions is called a population member, and the collection of population members is known as the population or as a generation. A population member is made up of individual elements known as chromosomes [16].

Chromosomes are the elements that help to distinguish one population member from another. In a binary population member, a "0" means the value to which that chromosome is mapped is not used, while a "1" indicates the value is used.

Once the initial population is created by randomly selecting all of the chromosomes of every population member, all of the population members are evaluated based on some measure of fitness. This fitness calculation is used to rank the population members and is used to help create the next generation. In discussions about genetic algorithms, the opposite of fitness: error, is usually mentioned. As the error of a population member increases, the fitness of the population member decreases. The error of the genetic algorithm is also referred to as the cost function. The cost function computes the error that the population member has, and can include any of the error metrics discussed in Section 2.3, or another error metric the user decides to use. The cost function can also include additional terms to help guide the search space towards an area that satisfies additional criteria, similar to regularization techniques and Equation 2.1. The next generation is created by a combination of elitism, crossover, and mutation [16]. Figure 2.4 suggests two different generations and how the second is produced from the first. A general description of elitism, crossover, and mutation follows, while a detailed description of how our genetic algorithm implements each one can be found in Chapter 3. For more information on genetic algorithms, refer to [14].

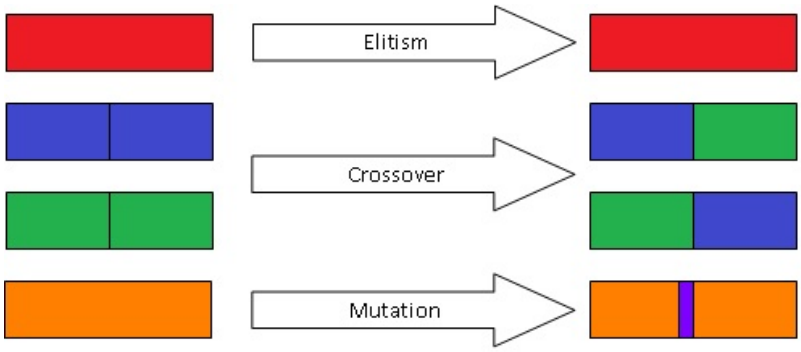

Figure 2.4: How a genetic algorithm creates a new generation

Elitism is the action of taking the best performing population members of the previous generation and allowing those same population members to be evaluated again in the new generation. The second type of action is crossover. Crossover takes two different population members and at a random chromosome switches the two different population members. The third type of action is mutation. Mutation takes a population member and randomly switches a chromosome. In a binary population member, a "0" is changed to a "1," and a "1" is changed to a "0."

After many generations, the most fit population members and chromosomes are spread through the entire population, and the genetic algorithm has determined a good solution to the problem. A genetic algorithm stops when the threshold for generations is reached, or the best fitness of the population does not change after a set number of generations [1, 8, 43].

A genetic algorithm has a few advantages over a normal search algorithm. A genetic algorithm does not have to know the exact problem that it is trying to solve.

The genetic algorithm is only searching for a population member that maximizes the fitness value, so it does not need to know any of the inner workings of the problem or even of the fitness calculations [16]. A genetic algorithm is also good at performing global searches without having to compute the fitness of every possible combination of chromosomes to find a well performing population member. A genetic algorithm accomplishes this by using generations to alter the search space and mostly looking in areas that have been determined to produce good results.

A genetic algorithm also has some disadvantages. A genetic algorithm takes a lot of computing power and time to converge to the globally optimal solution. The best performing population members are constantly being evaluated even though their fitness might not change [33], as they are constantly being passed from one generation to the next through elitism. To be effective, a genetic algorithm normally needs a population size in the hundreds and to be allowed to run for thousands of generations. A genetic algorithm also struggles to find good solutions for constrained problems [1, 16]. However, if the disadvantages mentioned can be addressed, the evolutionary programming approach discussed next may be able to produce an ensemble model that is more accurate than GasDay's current Dynamic Post Processor.

A genetic algorithm with some alterations will be explained in Chapter 3. In the next section, an introduction of evolutionary programming will be presented that is based on the general genetic algorithm presented here.

2.5 Evolutionary Programming

Another way to produce an ensemble forecast is to use an evolutionary programming approach, which was first used by Fogel in 1960 [13]. Evolutionary programming is a coding strategy by which an evolutionary programming engine, through a genetic algorithm, creates an evolutionary programming output, a computer program that answers the desired problem instead of the person writing the code. A more advanced approach is discussed in [21], where Koza *et al.* discuss the achievements of evolutionary programming and what classifies as true evolutionary programming. In their view, the evolutionary program has to be able to reuse and alter self-created sub-functions, iterations, loops, recursions, and executable code. Our evolutionary programming engine discussed in Chapter 3, takes a more limited, more targeted approach. Its' target program is confined to a template. Koza *et al.* state that the evolutionary program has to produce human-competitive results with minimum human intervention. They discuss this as a ratio of the knowledge of the program divided by the knowledge of the human coders. In a program with little human interaction, the program is only given the

basic elements of what it is trying to solve, along with a fitness calculation. This code also has to be robust and be able to work in multiple disciplines and on different projects with little human altering of the code.

In [21] they talked about evolutionary programming for circuit design. They discuss how the evolutionary program should be given only basic elements of resistance, inductance, and capacitance, and the evolutionary program should determine automatically the topology and the component values based on the given fitness calculation. The more automated the process is, the better it is at producing results, as it will not be influenced by well–known beliefs and practices of how something should be done. With this approach, the final output of the evolutionary programming engine has been able to produce results that are competitive with human results and duplicate results of 20^{th} and 21^{st} century patented inventions along with patentable new inventions it has created itself.

A specific implementation of evolutionary programming was used by Roebber [29] to help forecast temperature and electrical demand. The process selects randomly from an input set: variables and comparisons, to produce and evaluate an if–statement. An example of this is

$$\text{IF } (var1\ O_r\ var2) \text{ then}$$

$$\delta = (c1 \times var3)\ O\ (c2 \times var4)\ O\ (c3 \times var5). \tag{2.7}$$

The same number of terms was also used by Sathyanarayan, Birru, and Chellapilla [31] but they did not use an if–statement condition. The elements of the if–statement are

- O_r, a relation operator,
- $var1, \ldots, var5$, all of the inputs,
- O, a mathematical operator (either addition or multiplication), and
- $c1, \ldots, c3$, constants.

Once all of the programming statements are evaluated and averaged based on the number of statements that were true, a fitness calculation is used to evaluate how well each member of the population performed in forecasting the desired value or values if used on time–series data. Roebber used as his fitness calculation Mean Absolute Error. By allowing the well–performing population members to continue to be evaluated in a future generation, and altering bad population members, the genetic algorithm is providing a mechanism that allows the population members not to get stuck in a local minimum [31]. A 3–D representation of this optimization is shown in Figure 2.5. As the error metric used to determine fit decreases to an eventual minimum, the fitness of the algorithms approaches a maximum peak. Next, we will discuss the ensemble model that is currently implemented in GasDay.

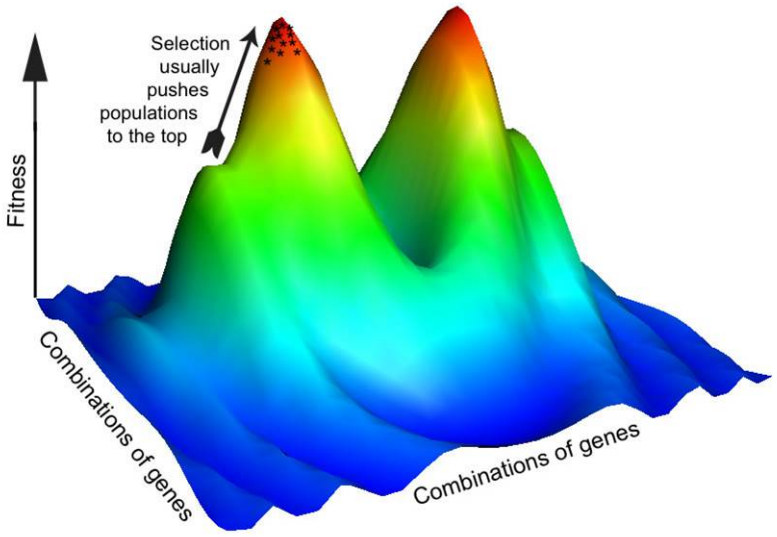

Figure 2.5: 3–D plot representation of evolutionary programming fitness [30]

2.6 Current Ensemble Model

The GasDay ensemble model, known as the Dynamic Post Processor, is used to take individual component models and combine them to create a single forecast for the amount of gas a Local Distribution Company is going to use on a given day. This forecast is a combination of either two or four individual component models, and represents the best estimate that GasDay is able to produce. The parameters inside of the Dynamic Post Processor are allowed to vary daily, and these variations allow the Dynamic Post Processor to respond to changing conditions in an operating area to track the gas demand of a nonstationary customer base. The produced value ultimately is used by a Local Distribution Company to assist them in determining how much gas they need to nominate.

The ensemble model was first created in 1995. Since then, the ensemble model has gone through numerous changes to allow it to create a more accurate estimate. The ensemble model currently uses the forecasts of the two or four different models GasDay uses. GasDay forecasts for time horizons of up to eight days, with each time horizon forecast being independent from the rest. For this thesis, focus will be limited to time horizon 0, but the process described could be extended to the other seven days for which GasDay forecasts. To simplify the explanations in this section, only two input models are considered for the rest of this section. However, the ideas presented are expanded to all two or four of the base component models that GasDay uses in the application that is sent to GasDay's licensed customers. Figure 2.6 shows the ensemble model combining a Linear Regression (LR) model and an Artificial Neural Network (ANN) model. When GasDay is run, both the Linear Regression model and the Artificial Neural Network model independently produce a forecast for the desired day.

The Dynamic Post Processor can be broken up into two different sections, a tuner and a combiner. The tuner works to improve the individual component estimates, while the combiner determines the percentage that each base component model contributes to the overall estimate.

The tuner adjusts the individual forecasts based on how well they did at forecasting the natural gas demand from two days ago. Values from two days ago

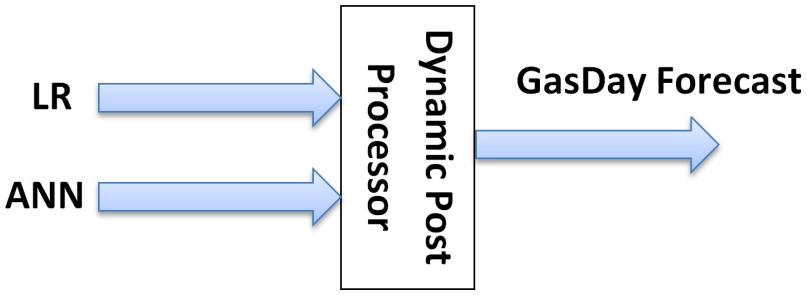

Figure 2.6: GasDay ensemble model for only two different component models

are used because GasDay is normally run at about 6 A.M. Central time. The national gas day starts at 9 A.M. Central time, so distribution companies do not have data for yesterday, as there is still about three hours left in yesterday's gas day.

The combiner takes the tuned individual component estimates and combines them based on how well they have been doing recently. If an individual component model becomes very good at forecasting the actual flow, that component model will contribute more to the overall estimate than if the component model recently has not been performing well. A graph of how the component weights changed over time from September 2010 through May 2013 for time horizon 0, is shown in Figure 2.7 for two component models.

The combination of the tuning of the individual component models and altering how the individual component models are combined, allows the Dynamic Post Processor to respond to changing characteristics in a Local Distribution Company's customer's usage. This section will be useful in Chapters 4 and 5, as the

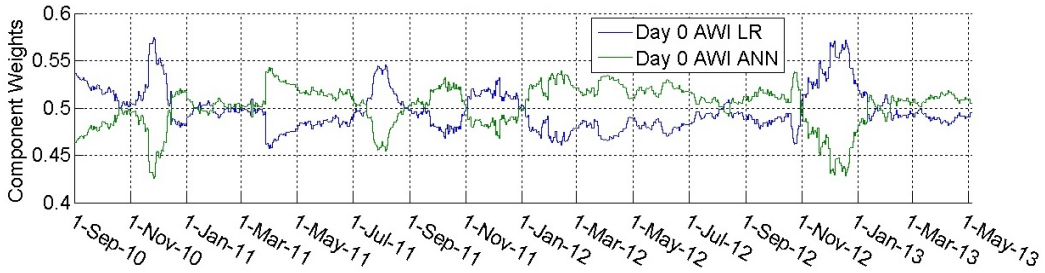

Figure 2.7: Component weights for time horizon 0 for two component models

created evolutionary programming ensemble model forecasts will be compared to the forecasts that were produced by the Dynamic Post Processor.

2.7 Conclusion

This chapter gave an introduction to different ensemble forecasting techniques and genetic algorithms. The error metrics presented here will reappear in Chapters 4 and 5 when the results of the evolutionary program are presented and compared to the current Dynamic Post Processor. Evolutionary programming was also introduced and will be further explained in the following chapter. This chapter also described the Dynamic Post Processor ensemble forecaster that is currently used in GasDay.

Evolutionary programming has been used in a wide range of applications that include weather forecasting, circuit designs, and other patentable inventions. One area where no research could be found is in natural gas forecasting. This area is what this thesis will try to fill, by showing how evolutionary programming

potentially can be helpful, not only in forecasting natural gas demand, but also in improving the accuracy of GasDay's estimates for our licensed Local Distribution Companies.

CHAPTER 3

EVOLUTIONARY PROGRAMMING APPLIED TO NATURAL GAS FORECASTING

This chapter gives a detailed description of the specific contributions of this thesis. It discusses both the genetic algorithm that is used by the evolutionary program engine to find the best combination of constants, as well as inputs for the evolutionary programming ensemble model design. This chapter also gives a description of the different inputs that could be used by the evolutionary programming engine and the evolutionary programming ensemble model design.

3.1 Rationale for this Work

As stated in Chapter 1, there is a need for accurate natural gas forecasts due to the margin of errors that Local Distribution Companies are allowed by the transportation companies and the repercussions that are possible if they do not have enough supply to meet demand. With these ideas in mind, there is always a need for a better forecast.

In Chapter 1, we also outlined a few requirements that a new ensemble model must meet. There are also additional requirements for GasDay and the students who will be working with the created ensemble model and MATLAB code.

- The ensemble model produces a forecast that is close to the actual flow used by the Local Distribution Company's customers,
- responds quickly to changes in natural gas load, and
- the MATLAB code must be understandable and maintainable to GasDay students, personnel, and Local Distribution Companies.

We discuss each in turn.

A new model must produce reasonably accurate and reliable forecasts. This is the reason Local Distribution Companies rely on GasDay to help supplement their in–house forecasters, due to the margin of errors that Local Distribution Companies are allowed by the transportation companies and the repercussions that are possible if they do not have enough supply to meet demand. With this idea in mind, Local Distribution Companies are always trying to be as accurate as possible with their gas forecasts.

A new ensemble model must be able to respond to changing loads without external assistance. GasDay would like to train the model for each of our Local

Distribution Companies only once a year and ship it to our customers, so the individual models have to be able to track changes. Every winter is different from previous ones, so there has to be capability in the created model to account for these changes, without intervention from a person. In at least one Local Distribution Company operating area, the Fall 2012 – Spring 2013 heating season had a 5% increase in customer use from the previous heating season given the same temperature. Figure 3.1 compares the flows from one operating area from December of 2011 to December of 2012, given a specific heating degree value. Not only is there an increase in the base load for this operating area shown by the shift in the y–intercept, but there is also a slight increase in the use per heating degree–day with the trend lines having slightly different slopes.

This does have some limitations, however. GasDay wants the ensemble model to be able to respond to changes in loads, but GasDay also wants to make it resistant to bad data, which is a common occurrence with flow data being entered daily. If the ensemble model were allowed to track changing conditions too fast, a single bad data point would have great consequences, while allowing the ensemble model to track changes too slowly means it would fail to adapt adequately to changing conditions. Figure 3.2 shows the errors of the Dynamic Post Processor if it were allowed to respond faster to changing conditions by comparing the percent error between the result of the current Dynamic Post Processor, and the results the

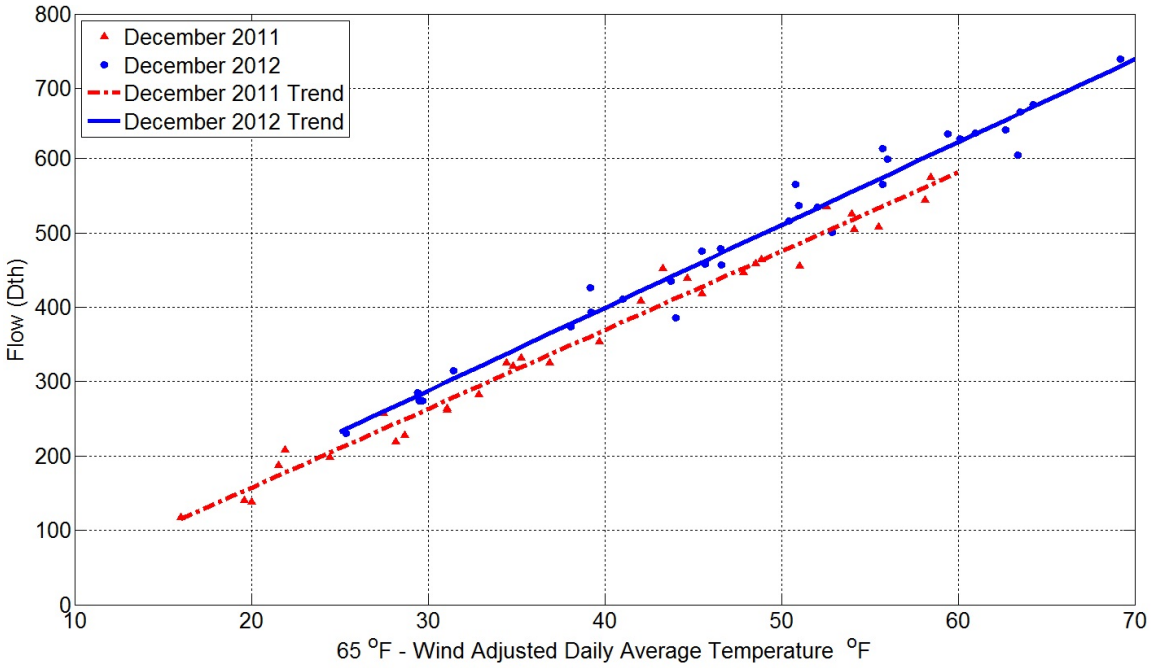

Figure 3.1: Scaled gas flow for two different years

Dynamic Post Processor would have produced if it were allowed to respond faster to changing conditions for a given operating area.

The implementation of the evolutionary programming engine and ensemble model in MATLAB must be able to be easily understood. GasDay is a research lab in a university, so the turnover of students is very high. Having well–written code limits the training required for new students, and they are more quickly able to contribute meaningful changes to new and existing code. Well–written code is also able to be reused by different functions and allows a compiler to optimize the execution of the code based on the ability of the machine on which the code is running [7]. Local Distribution Companies also have to present their gas forecasting

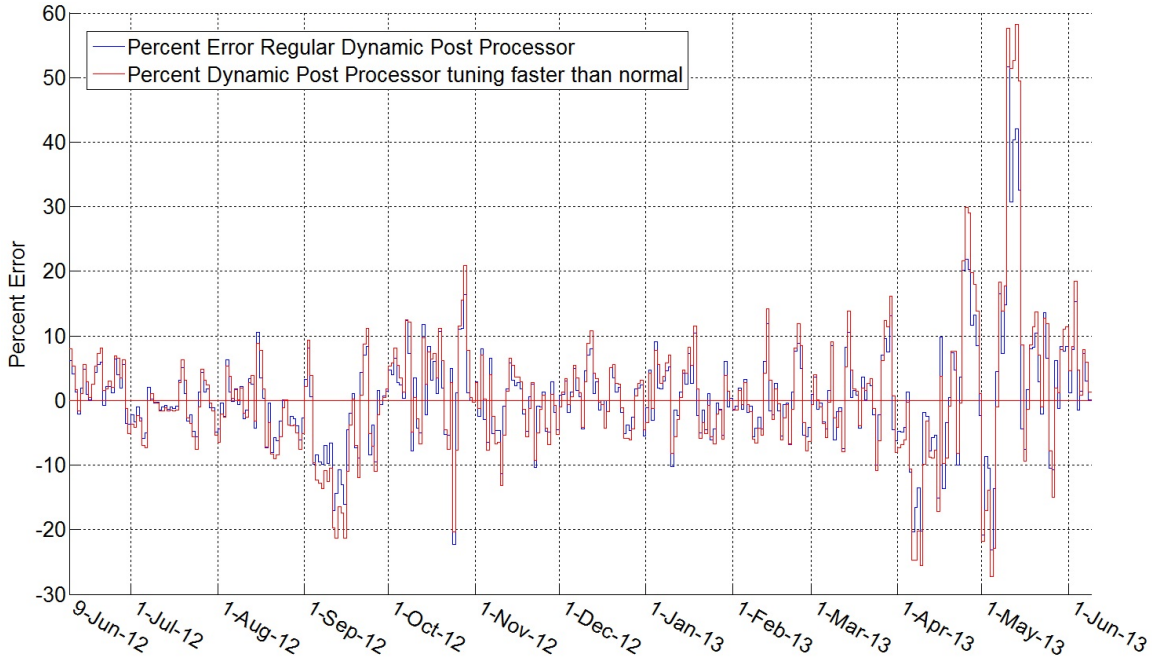

Figure 3.2: Percent error for the current Dynamic Post Processor and when the Dynamic Post Processor is allowed to tune faster based on the calculated error

techniques to the state to make sure their approach is acceptable [2]. If the created ensemble model cannot be explained to state regulators, there is a problem, as they have to approve the approach in the interest of the general public.

With these requirements in mind, work was begun on the evolutionary program engine and ensemble model to replace the current Dynamic Post Processor. An explanation of the code for both the evolutionary program engine, as well as the evolutionary programming ensemble model that is formed by the evolutionary program engine, can be found in the following section.

3.2 Evolutionary Programming Engine and the Evolutionary Programming Ensemble Model

The evolutionary program can be broken into three main categories: inputs into the evolutionary programming engine and ensemble model, output of the evolutionary programming engine, and the evolutionary programming engine itself. A diagram showing the evolutionary programming engine is shown in Figure 3.3. Each piece is discussed in the following sections.

3.2.1 Inputs into the Evolutionary Programming Engine and Ensemble Model

The inputs into the evolutionary program engine as well as the evolutionary programming ensemble model were determined by what would be important for an ensemble model that could be used for natural gas forecasting. The type of variables into both the evolutionary programming engine and the ensemble model are the same, except that the inputs into the evolutionary programming engine are year–long time–series values, while the data entered into the evolutionary programming ensemble model are the values for only 10 days. Not all of the inputs into the evolutionary programming engine or ensemble model are going to be used, but based on domain knowledge and past experience, a reasonable set of inputs were determined that had a chance to be important. Due to the uncertainty of what values are the best at forecasting the natural gas demand, all of the possible values

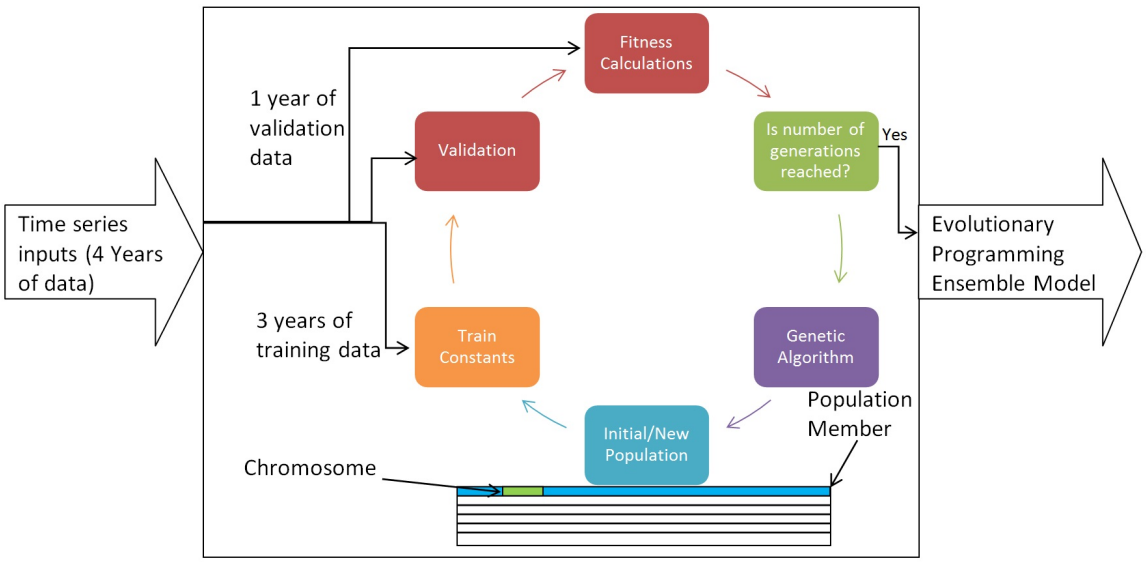

Figure 3.3: Overview of the evolutionary programming engine

used by the evolutionary programming engine also have to be given as inputs into the evolutionary programming ensemble model. Table 3.1 lists the possible inputs.

The Recently Tuned Error is computed using

$$\text{Recently Tuned Error} = \text{Previous Error} \times \lambda + e_{2 \text{ days ago}} \times (1 - \lambda), \qquad (3.1)$$

where $e_{2 \text{ days ago}}$ is the error that was computed using values from two days ago, and λ is equal to 0.99 [22, 38]. This value results in a daily error having a half–life of about two months as shown in Figure 3.4. This calculation weights the errors more heavily for recent days than days further back.

Table 3.1: Inputs in the evolutionary program and its output

Input	**Description**
Linear Regression	There are 10 days of Linear Regression estimates. The 10 estimates include the estimate for today as well as the estimates for the previous 9 days.
Artificial Neural Network	There are 10 days of Artificial Neural Network estimates. The 10 estimates include the estimate for today as well as the estimates for the previous 9 days.
Actual Flows	There are 8 days of actual flow because at 6:00 A.M. Central time when most utilities run GasDay, they have not completed the previous gas day that runs from 9:00 A.M. to 9:00 A.M. Central time.
Temperature	There are two days of forecast temperatures and 8 days of actual temperatures. The two days of forecast temperatures occur because at 6:00 A.M. when most utilities run GasDay, they have not completed the previous gas day that runs from 9:00 A.M. to 9:00 A.M. Central time.
Wind	There are two days of forecast wind speeds and 8 days of actual wind speeds. The two days of forecast wind speeds occur because at 6:00 A.M. when most utilities run GasDay, they have not completed the previous gas day that runs from 9:00 A.M. to 9:00 A.M. Central time.
Date code	The date code for the day, which is also used to compute the Day of the Week and Day of the Year.
Recently Tuned Error	This value is computed in the ensemble model by using the forecast estimate from two days ago and the actual flow from two days ago.
Moving Average of Previous Flow	Average of the 8 days of previous flow values that are available.
Additional Weather Input–Linear Regression	There are 10 days of Additional Weather Input–Linear Regression estimates. The 10 estimates include the estimate for today as well as the estimates for the previous 9 days.
Additional Weather Input–Artificial Neural Network	There are 10 days of Additional Weather Input–Artificial Neural Network estimates. The 10 estimates include the estimate for today as well as the estimates for the previous 9 days.

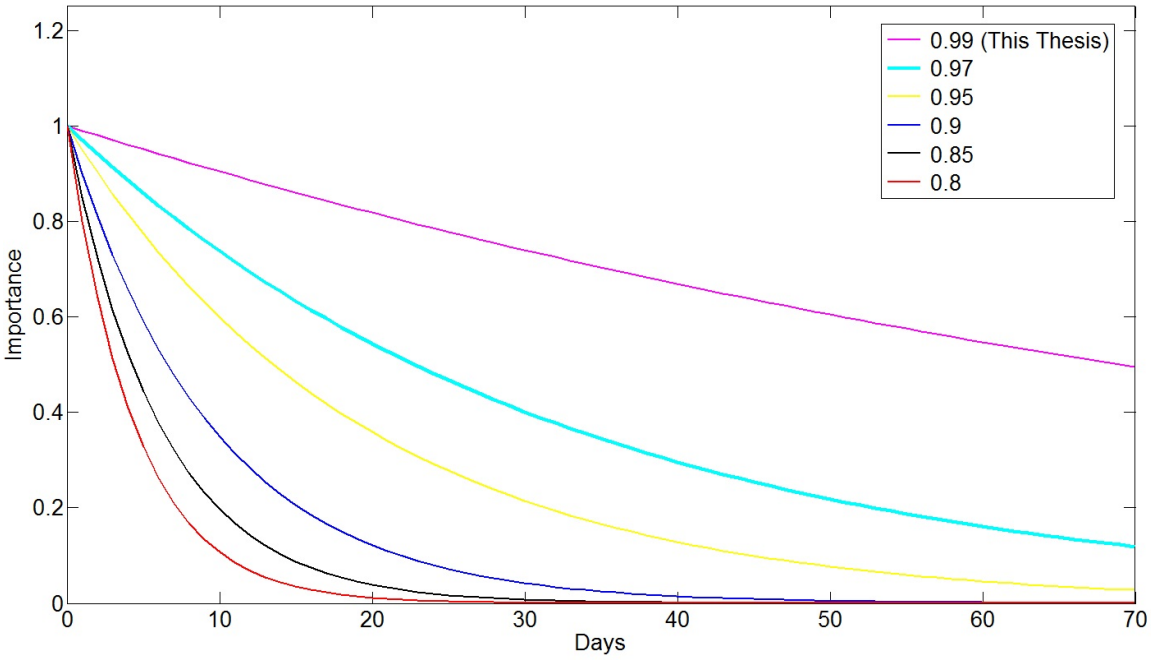

Figure 3.4: Day 0 error for 70 days with different forgetting factors

A visual representation of all of the raw inputs is shown in Figure 3.5. The diagram shows what inputs come from which specific day. All of the actual values that are used, are coming from at least two days ago because when GasDay is run, actual data for today and yesterday is not yet available.

Figure 3.5: Timeline showing what days all of the raw data inputs are coming from

These inputs are aligned into a vector array every day in which the evolutionary programming ensemble model is expected to produce an estimate so all of the inputs can be referenced simply by changing a number in a chromosome of a population member to something else. This allows crossover and mutation to occur easily, as the chromosomes do not have to reference the individual input arrays, which would greatly increase the complexity of the code and the ensemble model. Figure 3.6 shows how the evolutionary programming ensemble model would operate every day for time horizon 0, but the same process applies to the other time horizons that GasDay forecasts. For the other time horizons, the inputs shown in Figure 3.5 would change, as different values would be used depending on the time horizon that is being forecast.

3.2.2 Output of the Evolutionary Programming Engine

The evolutionary programming engine ultimately must produce MATLAB code for an ensemble model. We refer to this ensemble model as our evolutionary programming ensemble model to distinguish it from the evolutionary programming engine itself and from the current Dynamic Post Processor. This evolutionary programming ensemble model has to adhere to the limitations and specifications that were discussed in Section 3.1. The created ensemble model eventually may be

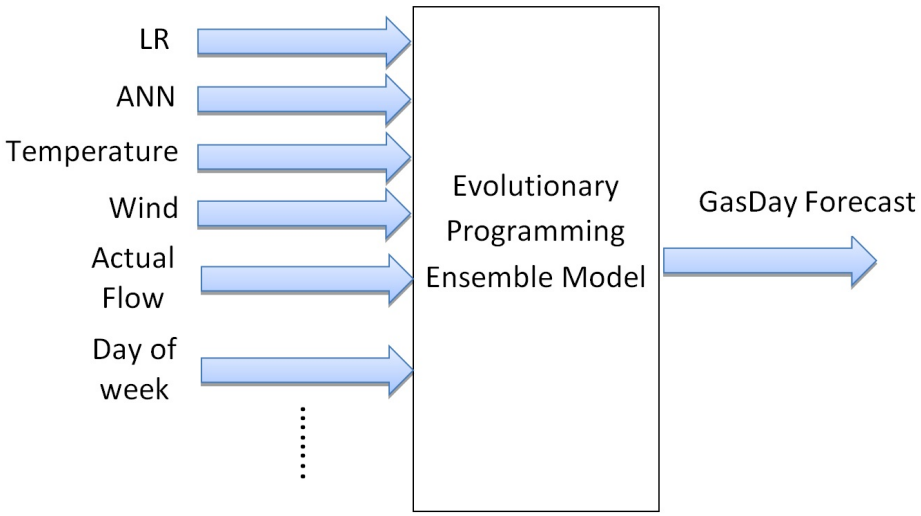

Figure 3.6: Proposed daily process at a Local Distribution Company

coded into C++ and shipped to Local Distribution Companies to run as part of GasDay and will use the inputs shown in Table 3.1.

Due to the process used to determine the best combination of the possible inputs, which is explained more in the next section, all of the values have to be given to the evolutionary programming ensemble model because the GasDay application will not know ahead of time which inputs were deemed important by the evolutionary programming engine.

The evolutionary programming ensemble model will have a few lines of code making sure certain variables are set to the appropriate values and are in the correct form for the evolutionary programming ensemble model. The evolutionary programming ensemble model itself will have two different types of statements. One statement is there to make sure the application is always able to produce an

estimate, regardless of how extreme the weather is or if the weather combination has never been seen before, i.e., design day conditions. The other statements will be designed as if–statements. The if–statements are similar to those used by Roebber [30], but there were some modifications based on testing different if–statement designs. An example of the if–statement that was used in Chapters 4 and 5 is

$$\text{IF } (var1 < c1 \times var2) \text{ then}$$

$$\delta = (c2 \times var3)\ O\ (c3 \times var4)\ O\ (c4 \times var5)\ O\ (c5 \times var6). \quad (3.2)$$

The elements of the if–statement are

- $var1, \ldots, var6$, all of the inputs,
- O, a mathematical operator (either addition or multiplication), and
- $c1, \ldots, c5$, constants.

A few lines of the actual code from an ensemble model are shown in Listing 3.1.

```
ŝ = c(1) * v(10) + c(2) * v(20) + c(3) * v(63) + c(4) * v(73) + c(5);
if (v(01) < c(6) * v(52))
    ŝ = ŝ + ((c(07) * v(58)) + (c(08) * v(42)) * (c(09) * v(02)) + (c(10) * v(01)));
    count = count + 1;
end;
if (v(24) < c(11) * v(65))
    ŝ = ŝ + ((c(12) * v(56)) + (c(13) * v(57)) * (c(14) * v(07)) * (c(15) * v(23)));
    count = count + 1;
end;
```

Listing 3.1: Example of evolutionary programming ensemble model if–statements

In Listing 3.1,

- `v()'s` correspond to one of the possible inputs and are adjusted by the genetic algorithm,
- `c()'s` are the constants that are learned during the execution of the evolutionary programming engine on the specified training data. The values can be positive or negative to reduce the complexity of the problem compared to [32],
- $\widehat{s}$ is the total sum of all of the executed if–statements and the one unguarded statement, and
- `count` is the total number of if–statements executed.

The four component models, Linear Regression (LR), Artificial Neural Network (ANN), Additional Weather Inputs–Linear Regression (AWI–LR), and Additional Weather Inputs–Artificial Neural Network (AWI–ANN) that the current Dynamic Post Processor uses to produce the GasDay estimate, offer the best estimates of what the flow will be if all of the if–statements are false. The probability of this happening is low, but there is a chance, so precautions have to be taken to ensure a value is always produced. The four component models also are very good estimators in their own right. A graph showing the four individual component models plotted against the actual flow for an operating area for one year

is shown in Figure 3.7. The current Dynamic Post Processor takes these four models and improves upon their individual results, while the evolutionary programming ensemble model takes a weighted linear combination of these four component models and combines them with the bodies of the if–statements that were true to assemble the estimate for the given day.

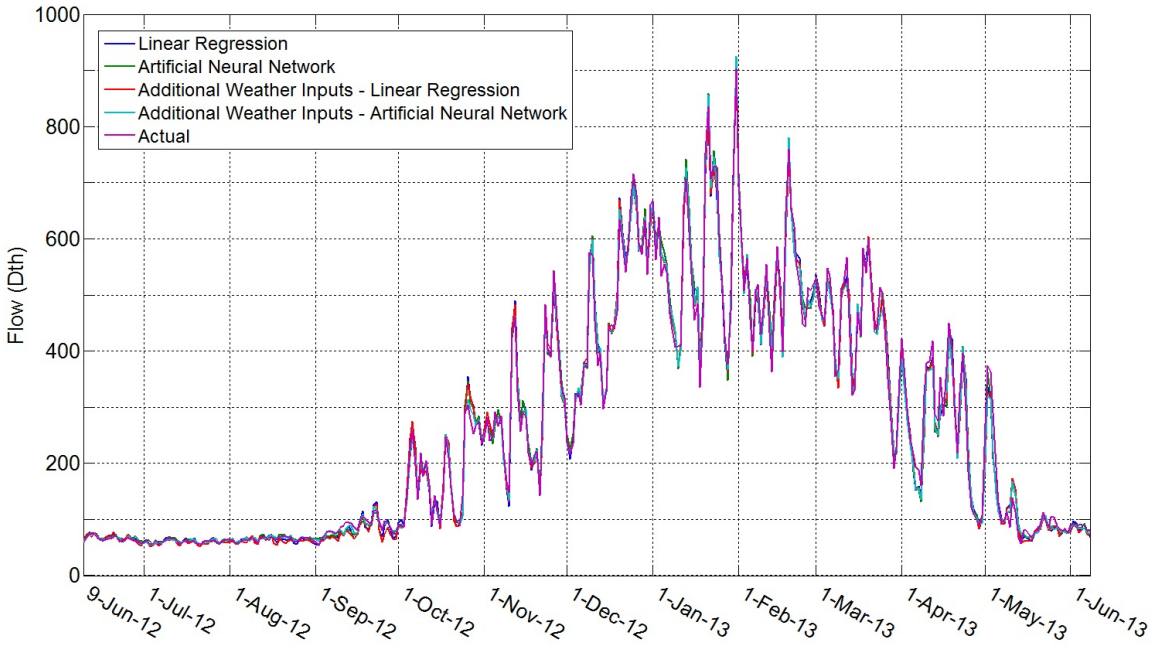

Figure 3.7: Time-series plot of the four individual component models and the reported flow

In Section 2.4, we discussed a binary genetic algorithm. Each chromosome in the array corresponds to specific inputs and lets the program know if it should or should not use that specific input. In this genetic algorithm, each chromosome is used by a specific location inside the final output design. Each chromosome can be a reference to any of the possible inputs. Each input is not unique, so there is no

limit to the number of times that a specific input can be used. In Listing 3.1, all of the numbers inside of the `v()'s` are a chromosome of the population member.

3.2.3 Evolutionary Programming Engine

The process to pick the ensemble model that will ultimately be working at a Local Distribution Company is a two–part process: training and validation of the constants of the different possible ensemble models and a genetic algorithm to choose possible ensemble models for training and validation.

Training is the process of determining the right values for the constants to be used with the inputs selected by the genetic algorithm. The procedure involves using MATLAB's `fminunc` command by the evolutionary programming engine to determine the optimal constants. This command uses initial values of the constants and estimates the derivative of the cost function [23]. Once this derivative is determined, `fminunc` changes the constants to reduce the cost function and estimates the derivative again. This process continues until `fminunc` reaches a point where the derivative is nearly zero, indicating that the cost function has reached a minimum, or that `fminunc` has reached the maximum number of iterations. The data used by the evolutionary programming engine is from previous years of data that were specified for training from the specific Local Distribution Company's database. In this thesis, three years of data, June 2008 – June 2011, were used to

estimate the optimal constants. To reduce the execution time of the evolutionary program, `fminunc` is limited to make only 1000 iterations of the optimization algorithm in a given generation per population member.

If the population member makes it through to the next generation, the evolutionary programming engine uses the previously determined constants to further fine–tune the constants to create a more accurate forecast. During elitism, all of the constants are used as the starting points for the next generation. If two population members are selected for crossover, the constants associated with the individual inputs or chromosomes are also crossed over and used in the next generation as the starting point for the evolutionary programming engine. By switching the constants with the variable chromosomes, we allow the evolutionary programming engine to have a better starting position than the default constants. It is possible that the constants are no longer that good when combined with the rest of the constants in the newly created population member, but they are better than having the constants start at the initial value. If an if–statement is selected for mutation, all of the constants associated with the chromosomes that were replaced are set back to their initial values. In this thesis, the constants were initialized to 0.1. The way the MATLAB code is written to determine the optimal constants, `fminunc` does not have to use all 1000 possible iterations. If MATLAB's `fminunc`

attempts various combinations, and it cannot find a lower error, the constants stay the same as the previous generation.

Validation data is used to help determine which population member is the best at forecasting the natural gas demand of a certain operating area, and which is just good at forecasting past natural gas demand. Validation data is not used in the training of the constants. The results of the validation data are used to adjust which population members the genetic algorithm uses to help determine which combination of inputs are the best at forecasting natural gas demand.

This is demonstrated in Figure 3.8, where the training error from the current generation is actually higher than the training error for the previous generation for the best performing population member. This is based on the population member with the lowest RMSE value on the training data, even though the evolutionary programming engine is supposed to improve the output at each generation. To demonstrate this, a trial run was conducted with a population size of 12, and the evolutionary programming engine was allowed to run for only 15 generations. In Figure 3.8, the error went up in generation 12 because the best performing population member, based on the lowest cost function value on the training data set, determined by RMSE, was not one of the best members at actually forecasting natural gas demand. A graph of the RMSE error of the population member that performed the best on the validation data and the corresponding training RMSE

value is shown in Figure 3.9. Unlike what occurred to the testing error, the validation error never went up across the generations as the best population member was always allowed to be passed to the next generation through elitism.

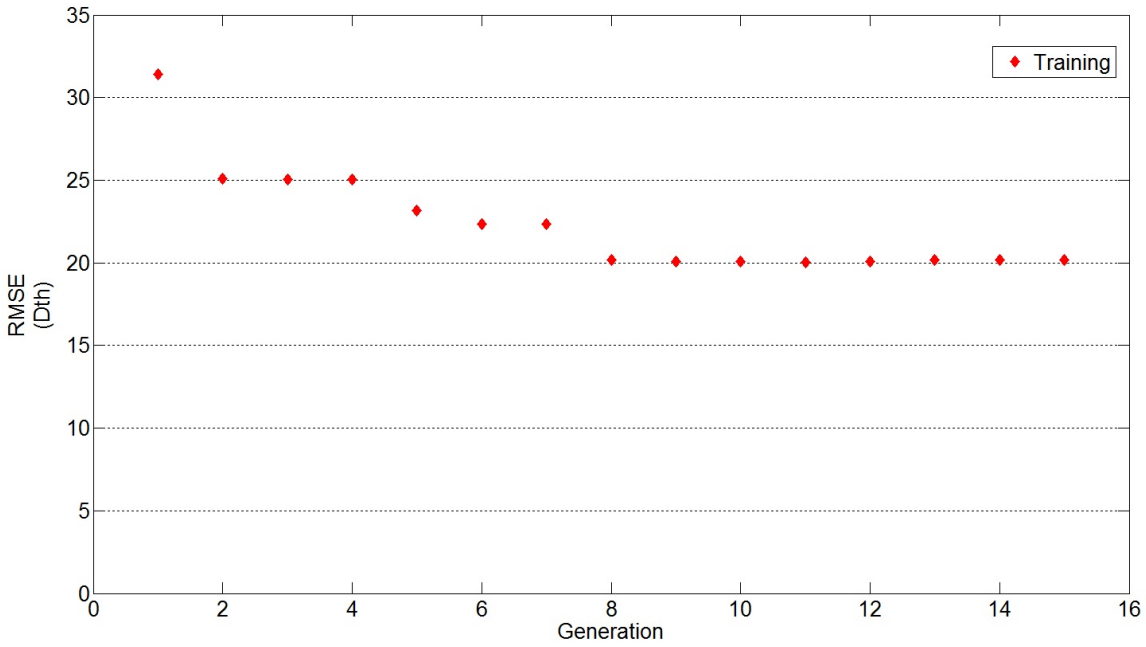

Figure 3.8: Evolutionary programming training error of the population member with the lowest error on the validation data

Testing data was used for analysis purposes only. The data used for testing was from June 2012–June 2013. To determine how well the evolutionary programming ensemble model was working, it was compared to the current Dynamic Post Processor. The results presented in the rest of this section were done only as a proof of concept. In Chapters 4 and 5, the number of generations and the size of the population were drastically increased to get the results that are analyzed against the Dynamic Post Processor. In Chapters 4 and 5, the evolutionary

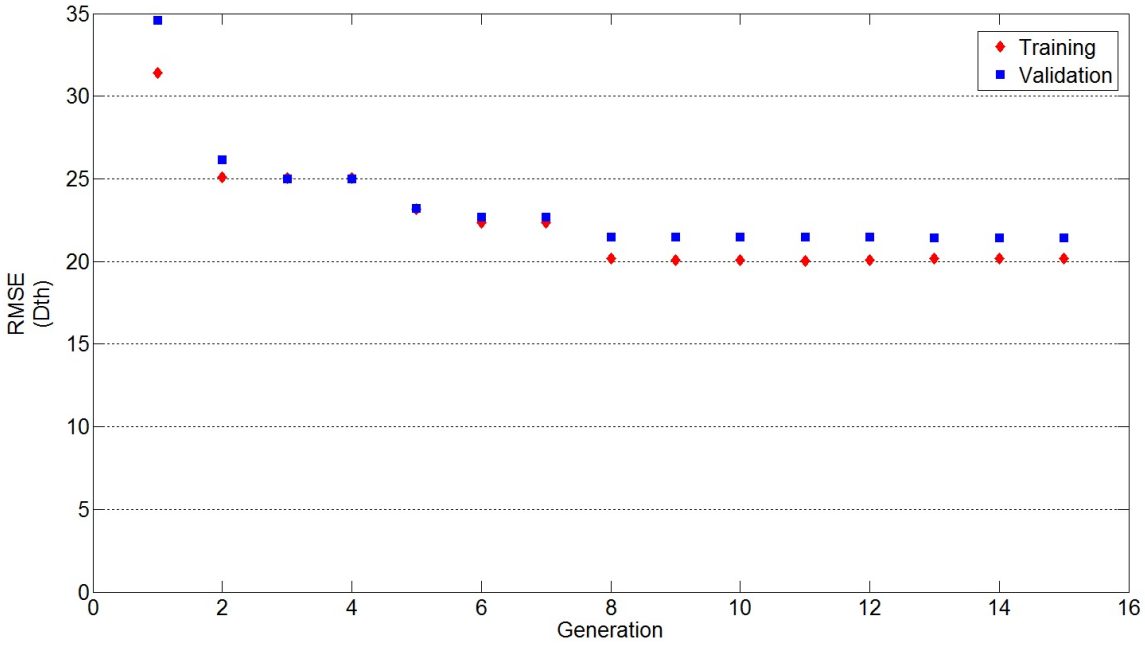

Figure 3.9: Evolutionary programming training and validation error of the best member of the population

programming engine was allowed to run for 100 generations with a population size of 800 members. The data was taken directly from a Local Distribution Company's database, so the data was unprocessed by GasDay and had all of the erroneous flow values that the current Dynamic Post Processor had to forecast with at the Local Distribution Company. This heating season also saw an increase in flow values from previous years, so it was a great season to compare how robust the evolutionary program engine was in producing a reliable ensemble model.

As stated previously, the genetic algorithm uses the validation error to help determine what inputs are important for forecasting. The genetic algorithm allows

the top 10% of the best members of the current generation to move on to the next generation and continue to be evaluated.

For this thesis, 40% of the best validation members are used for crossover, where the values and constants from two different population members are switched at a random point to create two different population members to be evaluated in the next generation. The point where the crossover occurs is random and constantly changes from one pair to another.

To create another generation of the same size, the genetic algorithm takes 50% of the most fit population members to create the same number of population members as the previous generations [30], and each if–statement has the possibility to be selected for mutation. Mutation occurs when one of the if–statements is replaced with another random set of chromosomes to try to find a combination that has a lower error and can guide the genetic algorithm to search in the space around that member of the population. Normally, only a single chromosome is selected and changed for mutation [16].

3.3 Small Scale Test

To demonstrate that the evolutionary program ensemble model produced a forecast that was more accurate than the current Dynamic Post Processor, a test was

conducted using the actual data from a Local Distribution Company as well as the estimates for the Dynamic Post Processor. The test consisted of an initial population size of 12 members and the evolutionary programming engine was allowed to run for 15 generations. The same data that GasDay had access to in the Fall of 2011 was used to train and validate the evolutionary programming population members. The evolutionary programming ensemble model that preformed the best on the validation data was then used to forecast the natural gas demand using the same uncleaned data that GasDay had access to during the heating season from the Local Distribution Company. Table 3.2 shows the difference between the current Dynamic Post Processor error and the error produced by the evolutionary programming ensemble model from the same example shown in Figures 3.8 and 3.9. The values in the table represent the month-long Root Mean Square Errors, RMSE, between the current Dynamic Post Processor and the evolutionary programming ensemble model estimates compared to the actual flow values from April 2012 – March 2013. Even using a small sample size of 12 population members and 15 generations, the evolutionary programming ensemble model was able to produce time horizon 0 forecasts that were better that the current Dynamic Post Processor in 8 of the 12 months.

This small sample was only used for testing the code to make sure that the values that it was calculating for errors were correct and there were no errors in the

Table 3.2: RMSE by month for the Dynamic Post Processor and the evolutionary programming ensemble model

Month	**Dynamic Post Processor**	**Evolutionary Programming Ensemble Model**
April	20.63	20.30
May	14.25	13.72
June	5.30	5.23
July	4.35	4.37
August	3.10	3.08
September	9.49	9.72
October	7.60	7.51
November	26.97	27.94
December	25.75	25.13
January	36.29	33.32
February	34.66	35.24
March	24.28	22.79

code. When this small sample experiment was finished, the estimates for every day in the testing set were copied and manually compared to the actual estimates to make sure that the values were the same. It was also used to see if there was any hope in having the evolutionary programming ensemble model compete with the current Dynamic Post Processor. These results are from the beginning of the process to determine if a completely different approach should have been pursued so in the end, the work of this thesis provided results that were better than what GasDay currently has implemented. In Chapter 4 and 5, results will be presented showing how well the evolutionary programming approach worked when it was allowed to have an initial population of 800 members and allowed to run for 100 generations.

3.4 Advancements to Roebber's Work with Evolutionary Programming

As mentioned in Chapter 2, our evolutionary programming engine and evolutionary programming ensemble model were inspired by research done by Roebber [30]. Roebber's research was used in a slightly different strategy from what was proposed in Chapter 2. His work used evolutionary programming to create many good ensemble models, which were then used to create an overall forecast, while GasDay wants to use the evolutionary programming approach to create a single ensemble model that can be shipped to our customers. After reading Roebber's work [30], it was determined that there was a possibility to expand on his work and use it in the forecasting of natural gas. Some of the advancements are

- what if no if–statements are executed, and
- duplicate population members.

We discuss each in turn.

A major item is how the evolutionary program handles no if–statements being executed. This would be a problem for Local Distribution Companies that rely on GasDay to help forecast the amount of natural gas needed for a certain day. If none of the if–statements were true, the forecast natural gas demand would be 0, which is never the case for an entire operating area. Dr. Roebber handled this by

making the if–statements opposites of each other to ensure some of them execute [30]. An example of this is shown in Listing 3.2.

IF $(V(12) \leq \triangle MOS)$ then...

IF $(V(12) > \triangle\, MOS)$ then...

Listing 3.2: Example of Roebber's code

Roebber uses both '>' and '<' comparators, but only one is needed as Var1 > Var2 is the same as Var2 < Var1. This method does solve the problem of not having any statements being true, but it can be made more efficient as every combination of variables in the if–statement need to have the opposite as well, resulting in twice as many statements overall than what will actually be executed on a single day. By limiting the number of possible comparators, the total number of different combinations that have to be compared is drastically reduced. Reducing the total number of combinations will reduce the complexity of the problem for the genetic algorithm and help it locate the best combination of chromosomes faster.

Roebber's work also does nothing to guard against duplicate population members. With many statements in each member of the population, there is the possibility that all of the statements are the same as another population member, but in a different order. While this will not affect the final answer determined by

the genetic algorithm, it drastically increases the total number of possible solutions and can make it harder for the genetic algorithm to converge.

3.5 Conclusion

This chapter described the evolutionary programming engine, and the code that was generated and used to produce the ensemble model that forecasts gas demand on a given day. The process presented here was only for Day 0, but can be repeated for the other time horizons that GasDay forecasts. It also discusses improvements that were completed compared to Roebber's work [29] on which this thesis was built. Chapter 4 will present the results for a number of different operating areas using two different evolutionary programming ensemble model designs. Chapter 5 will present some ensemble model designs that did not work well, while providing some insights into why they were not good at forecasting natural gas demand.

CHAPTER 4

QUALITY OF FORECASTS FROM THE EVOLUTIONARY PROGRAMMING ENSEMBLE MODEL

While working on the evolutionary programming ensemble model, different designs were tried and evaluated to see which produced the most accurate forecast results. While conducting these tests, different numbers of constants, variables, if–statements, comparators, and operating areas were used. Some were allowed to be varied by the genetic algorithm, and some were set based on domain knowledge.

The majority of this chapter will focus on the evolutionary programming ensemble model design that produced the most accurate forecast results in a three–way comparison of different ensemble model designs. These designs will be referred to as Design A, Design B, and Design C.

- Design A uses the average of a weighted linear combination of the four base component models and 10 if–statements.
- Design B uses a weighted linear combination of the four base component models, where the coefficients are determined by the average of five if–statements.
- Design C uses a weighted linear combination of the four base component models, where the coefficients are determined by the sum of five if–statements.

Ensemble model Design A was used to generate the results in the first half of this chapter, and its generated forecasts were more accurate than the current Dynamic Post Processor in our tests. The second half of this chapter will focus on ensemble model Design D, which produced results that were not as accurate as ensemble model Design A, but produced results that were more reasonable from a domain knowledge point of view. Ensemble model Design D is a weighted linear combination of the four base component models added to the average of 10 if–statements. Neither ensemble model Design A nor Design D were able to produce results that were statistically significant or reasonable to a level that would suggest that the current Dynamic Post Processor be replaced by the evolutionary programming ensemble model, but ensemble model Design A was able to perform better than the Dynamic Post Processor in our tests. The ensemble models also took 3–4 days of computing time to train on a 20–core cluster. The cluster is a heterogeneous collection of Intel Xeon and Intel i7 processors. Additional information on the cluster configuration can be found in Section 5.3. This amount of time is not reasonable if GasDay would have to create eight ensemble models for the eight different time horizons for all 170 operating areas that GasDay forecasts natural gas demand.

Table 4.1: RMSE values for the Dynamic Post Processor and ensemble model designs A, B, and C for four different operating areas from June 2012 – June 2013

		Evolutionary Programming Ensemble Model		
Area	**Dynamic Post Processor**	**Design B (average)**	**Design C (sum)**	**Design A**
1	18.69	80.21	31.21	17.95
2	22.57	75.28	24.71	X
3	31.66	95.08	41.68	X
4	23.85	48.69	28.74	24.58

4.1 Determination of the Most Accurate Design

Ensemble model Designs A, B, and C were compared to see which model produced the most accurate forecasts. The results of this comparison are in Table 4.1. This test was conducted with a population size of 200 members for 70 generations. Two of the model designs, Design B and Design C, use the generated if–statements to produce the coefficient values on a linear combination of the four base component models that GasDay uses:

$$\widehat{s} = \beta_1 + \beta_2 \times \text{LR} + \beta_3 \times \text{ANN} + \beta_4 \times \text{AWI-LR} + \beta_5 \times \text{AWI-ANN}. \tag{4.1}$$

By using an average, ensemble model Design B put more pressure on each statement to give an accurate result of what the coefficient should be, whereas using a summation with ensemble model Design C allowed more freedom. These two designs will be discussed in the following chapter due to their performance.

Ensemble model Design A, which will be discussed in the first half of this chapter, averages a weighted linear combination of the four base component models along with the generated if–statements:

$$\widehat{s} = \frac{\begin{array}{c}(\beta_1 + \beta_2 \times \text{LR} + \beta_3 \times \text{ANN} + \beta_4 \times \text{AWI-LR} + \beta_5 \times \text{AWI-ANN}) \\ + \sum \text{true if-statement bodies}\end{array}}{1 + \sum \text{true if-statements}}. \tag{4.2}$$

Ensemble model Design A used data from three different operating areas from three different Local Distribution Companies to generate a daily forecast. These forecasts were compared to the forecasts that were generated by the Dynamic Post Processor. The three operating areas will be referred to as Alpha, Bravo, and Charlie, respectively, and all of the data has been scaled to protect the identity of the Local Distribution Company. These operating areas are not the same areas that were presented in Table 4.1.

The equation used to scale the data was

$$\text{Scaled Value} = \frac{950}{\text{max value over the series}} \times \text{Value}. \tag{4.3}$$

This equation allowed the data to appear on a graph that had values from 0–1000 to create a clean graph when the flow values are graphed across the entire data set. This equation also hides the identity of the Local Distribution Company whose data

is being used in conjunction with our signed license agreements. If this work could be redone, this scaling would have taken place before the values were used by the evolutionary programming engine and ensemble model instead of altering all of the daily forecasts.

4.2 Evolutionary Programming Ensemble Model Design A

Ensemble model Design A, shown in Figure 4.1, was used to generate the results in the beginning part of this chapter and was a combination of 10 if–statements and one unguarded statement that was a weighted linear combination of the four base component models with a constant offset. The results for the comparison in operating areas Alpha, Bravo, and Charlie were obtained using a population size of 800 members, and the evolutionary programming engine was allowed to run for 100 generations. As will be discussed further in Section 5.3, a single ensemble model takes 3–4 days on a 20-core cluster for the correct combination of chromosomes to be determined and have all of the constants optimized. Ensemble model Design A requires 45 constants to be determined. All of the constants were determined by the evolutionary programming engine and `fminunc`. For a full description of how `fminunc` works, refer to Section 3.2.3. On any given day, the forecast value from evolutionary program ensemble model Design A could be the average of as many as 11 statements or as few as one. The statement that was a weighted linear

combination of the four component models makes sure that no matter what inputs are given to the evolutionary programming ensemble model, an estimate will always be produced, assuming there are not any problems in the rest of the process, either in GasDay or at the Local Distribution Company. Some common problems include network problems and incomplete data files.

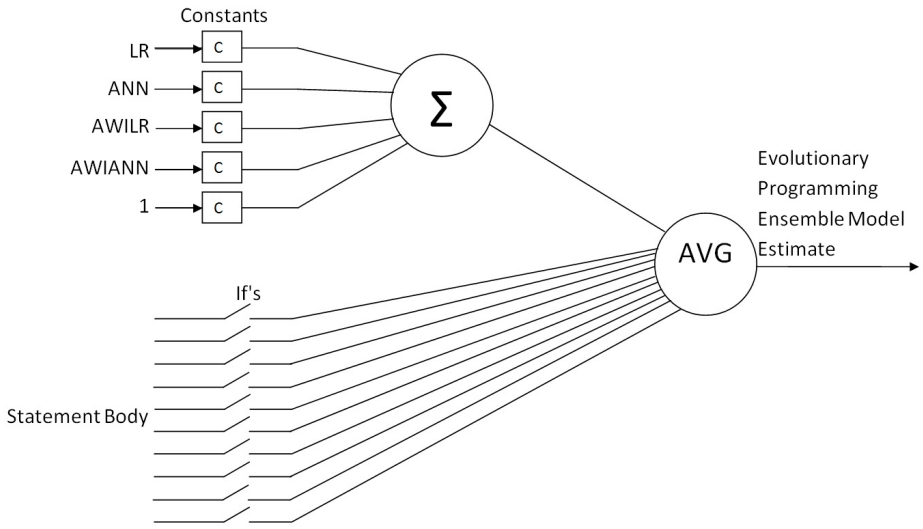

Figure 4.1: Evolutionary programming ensemble model Design A using one unguarded statement and 10 if–statements

Ensemble model Design A allows freedom in the constant values of the if–statement bodies to produce a wide range of possible results. For a detailed description of the if–statement bodies, see Section 3.2.2.

4.3 Comparing the Dynamic Post Processor and Evolutionary Programming Ensemble Model Design A

Once the genetic algorithm determined the chromosomes that resulted in the lowest error on the validation data, and the evolutionary programming engine had finished fine-tuning the optimized values for the constants throughout all of the generations, ensemble model Design A was evaluated on previously unseen test data from June 2012–June 2013. This type of testing is also referred to as out-of-sample testing or ex-ante testing [17]. This testing took place in the same manner in which the population members were evaluated during the training and validation process, but on data that the population member had never seen before. The evaluation of the model took place on a day–by–day basis. The forecast for each day was recorded, and at the end of the testing data, the RMSE value for the year was calculated. The RMSE value was also calculated for different combinations of days to provide a detailed view of how the evolutionary programming ensemble model performed. A few of these combinations include individual months and types of days. GasDay studies different types of days to determine how well GasDay performs on days that are unusual or days that have special characteristics that make producing a good forecast difficult. For additional discussions on the unusual days that GasDay considers, see [28, 40]. Table 4.2 shows the different types of days that GasDay focuses on and a description of each type of day. The “First Warm Days” and

"First Cold Days" are harder to forecast due to the uncertainty of how many people will use their furnaces and how many will open their windows. For a full description of how the training and validation was performed, see Section 3.2.3. In addition to the RMSE value, additional statistics were calculated to aid us in making an informed decision about the quality of ensemble model Design A's forecasts.

Most of the graphs presented in this chapter are in one of three different forms. One is a time–series graph that shows different values for the Dynamic Post Processor and the ensemble model design for the testing set. Another is a graph showing the errors (forecast - actual flows) decomposed by month, along with the total error for the entire period analyzed. The other type of graph shows the error decomposed based on the type of day that it is, along with the total error across all days. All three types of graphs are commonly used by GasDay to measure the performance of its models, and these are commonly included in reports that GasDay sends to Local Distribution Companies that license GasDay, to provide a detailed description of how GasDay performed for each of their operating areas.

4.4 Operating Area Alpha

The first operating area that was chosen for a comparison between the Dynamic Post Processor and ensemble model Design A was from a Local Distribution Company that had a very high year–over–year growth from the Fall 2011-Spring

Table 4.2: "Unusual" day types, as considered by GasDay

Label on graphs	**Description of the type of day**
All Days	All of the days in the testing set. Here, the days of comparison are from June 9, 2012, through June 8, 2013.
Coldest	The 5% coldest days in the testing set.
Colder than Normal	The 5% of the days in the testing set that were the coldest compared to the normal temperature on that day.
Warmer than Normal	The 5% of the heating days in the testing set that were the warmest compared the normal temperature on that day.
Windiest Heating Days	The 5% of the heating days in the testing set that were the windiest.
Colder Today than Yesterday	The 5% of the days in the testing set that have the largest drop in temperature from yesterday to today.
Warmer Today than Yesterday	The 5% of the days in the testing set that have the largest increase in temperature from yesterday to today.
First Cold Days	The first few days that have temperatures below 65°F after the summer months.
First Warm Days	The first few days that have temperatures above 65°F after the winter months.
High Humidity Heating Days	The 5% of heating days in the testing set that have the highest humidity.
Low Humidity Heating Days	The 5% of heating days in the testing set that have the lowest humidity.

2012 compared to the Fall 2012–Spring 2013 heating seasons. In Figure 4.2, the use per heating degree–day for the 2012-2013 heating season is much higher than the use per heating degree-day in the 2011–2012 heating season. For a given temperature, the flow for December 2012 is higher than the flow for December 2011, indicating that there was an increase in usage per heating degree-day.

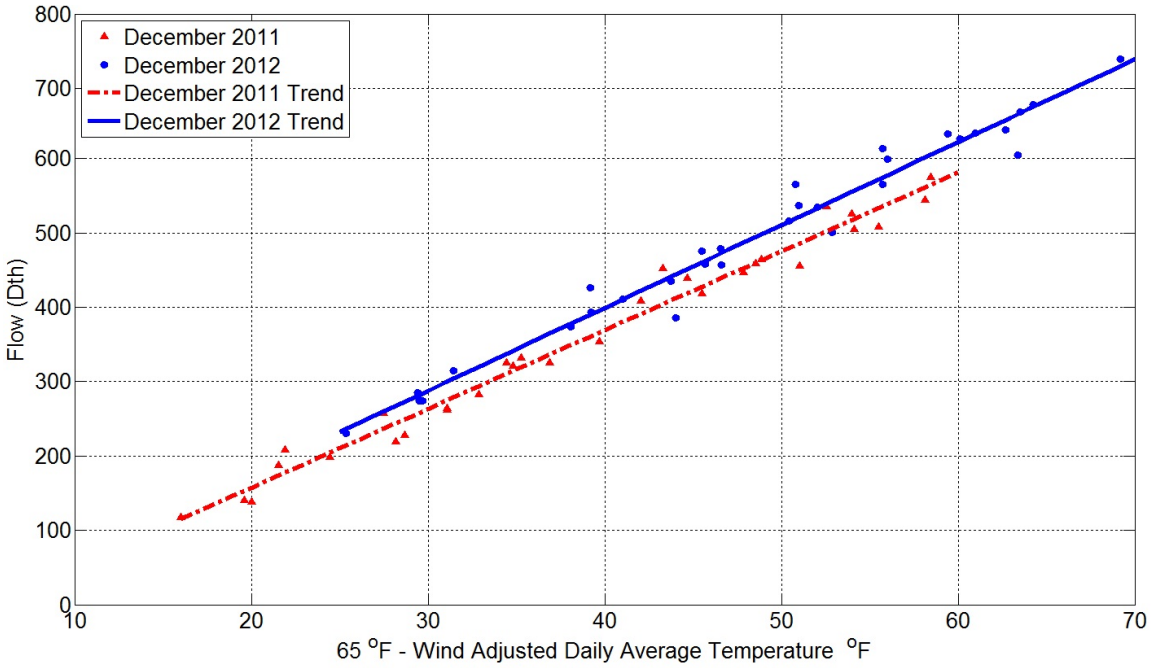

Figure 4.2: Scaled gas flow for two different years, operating area Alpha

In Figure 4.3, there are three graphs comparing the flow for the Local Distribution Company's database, the flow estimated from the Dynamic Post Processor, and the flow estimated from ensemble model Design A. The other two graphs in Figure 4.3 show a time–series plot of the errors of each forecasting approach compared to the flow values that were in the database for the Local Distribution Company. These values had all of the erroneous flows that might have been entered into GasDay on site, and GasDay did not clean the data before the comparisons were done. The graphs show that ensemble model Design A tracks the actual flow quite well, as there are no large errors present during the testing year compared to the Dynamic Post Processor.

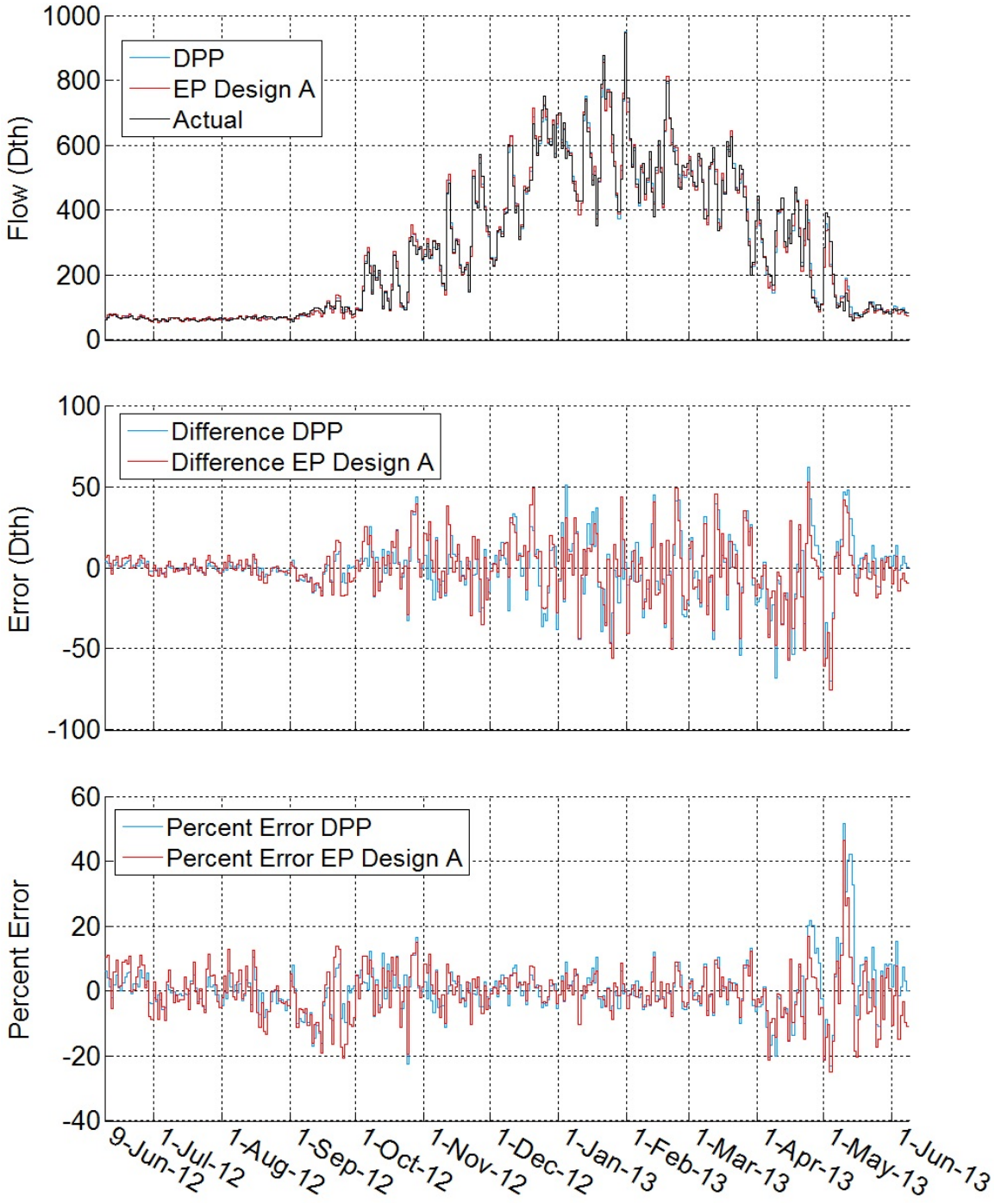

Figure 4.3: Time-series of operating area Alpha from June 2012 - June 2013

In Figure 4.4, there are again three graphs showing different error values, but they are decomposed by month for the same time–frame as shown in Figure 4.3. June is not included because it was split over two years, but the data for the partial months is included in the "All Days" section. Comparing both the Standard Error values (top graph) and the RMSE values (bottom graph) for the entire data set, ensemble model Design A forecasts were more accurate than the Dynamic Post Processor overall. For a description of these error metrics, see Section 2.3. Ensemble model Design A was not more accurate for every month, however. When looking at the Standard Error values, October and May had very similar results between the two models, while ensemble model Design A performed better in December, March, and April. When looking at the RMSE values, evolutionary program model Design A again had a lower error for "All Days," but it performed worse than the Dynamic Post Processor in the months of July, August, and November.

Figure 4.5 shows the same three error metrics as in Figure 4.4, but the values are decomposed into the types of days described in Table 4.2. The graph of the Standard Error values shows ensemble model Design A performed better than the Dynamic Post Processor on days that were "Warmer than Normal," "Warmer Today than Yesterday," and the "First Warm Days," but the evolutionary programming ensemble model was out-performed on the "Colder than Normal" days and on the days that were "Colder Today than Yesterday." However, when looking

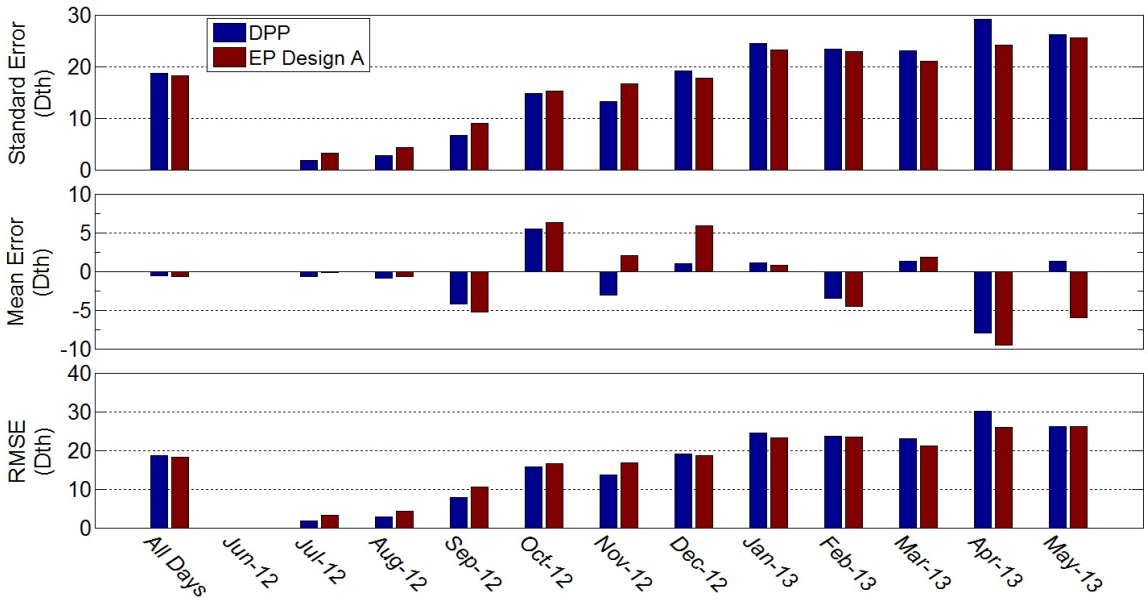

Figure 4.4: Operating area Alpha error decomposed by months

at the RMSE values, ensemble model Design A performed better compared to the Dynamic Post Processor on "All Days," along with every other type of day except for the "Colder than Normal" days and the "Windiest Heating Days."

Based on Figures 4.4 and 4.5, ensemble model Design A performed better than the current Dynamic Post Processor. However, as explained in Section 2.4 and Section 5.3 in the next chapter, the increase in accuracy is not enough to overcome the obvious problem of the amount of time that was needed to produce these results. The lack of a significant increase also can be shown by running a T-test on the data. Ensemble model Design A's results for "All Days" were not found to be statistically significant at conventional levels (5%).

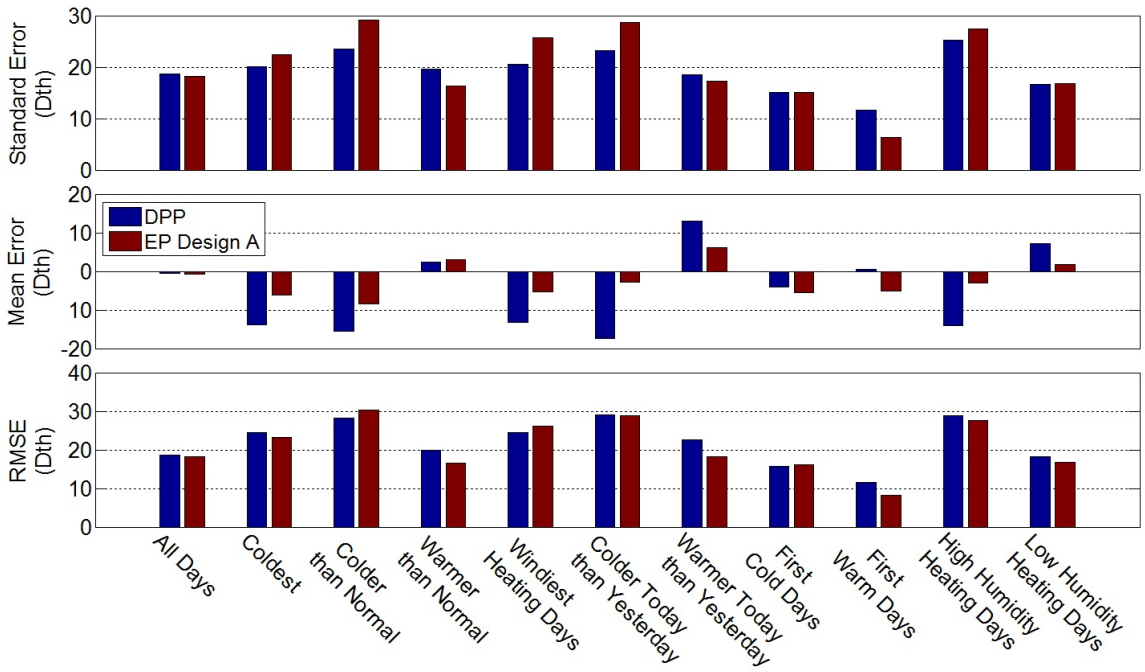

Figure 4.5: Operating area Alpha error decomposed by type of days

4.5 Operating Area Bravo

The second operating area that was considered was from a Local Distribution Company located in the northern part of the United States to see how well ensemble model Design A performs in an operating area whose consumption is very temperature dependent. Again, ensemble model Design A was determined by the evolutionary programming engine running for 100 generations with a population size of 800 members and took about three days on a 20-core cluster for the evolutionary programming engine to reach 100 generations.

In Figure 4.6, there are three graphs comparing the flow estimates from the Dynamic Post Processor, the flow estimates from ensemble model Design A, and the

flow values that were entered into the database by the Local Distribution Company. At the end of October, there is a huge spike by the Dynamic Post Processor compared to the flow value in the database. When we went back and looked at the data, the error came from the fact that both the Artificial Neural Network and the Additional Weather Inputs - Artificial Neural Network models had estimated that the flow would be twice what it actually was. This error is also present in the other two graphs. By looking at the bottom graph, percent error, it can be seen that the Dynamic Post Processor overestimated the actual flow by about 130%. Due to this percent error being above 100%, we know that the percent of these two models was more than 50% when combined by the Dynamic Post Processor, compared to the Linear Regression and Additional Weather Inputs - Linear Regression models. This is known because by doubling two of the estimates, the error should have only been 100% when all four of the models are combined equally as described in Section 2.6. When the models were re-evaluated at GasDay, this spike did not occur. Because the spike could not be duplicated, it was determined that this Local Distribution Company changed some data in the database but never re-fired the models. For this reason, the specific day with the exceptionally large Artificial Neural Network and the Additional Weather Inputs - Artificial Neural Network forecasts was removed from the testing set.

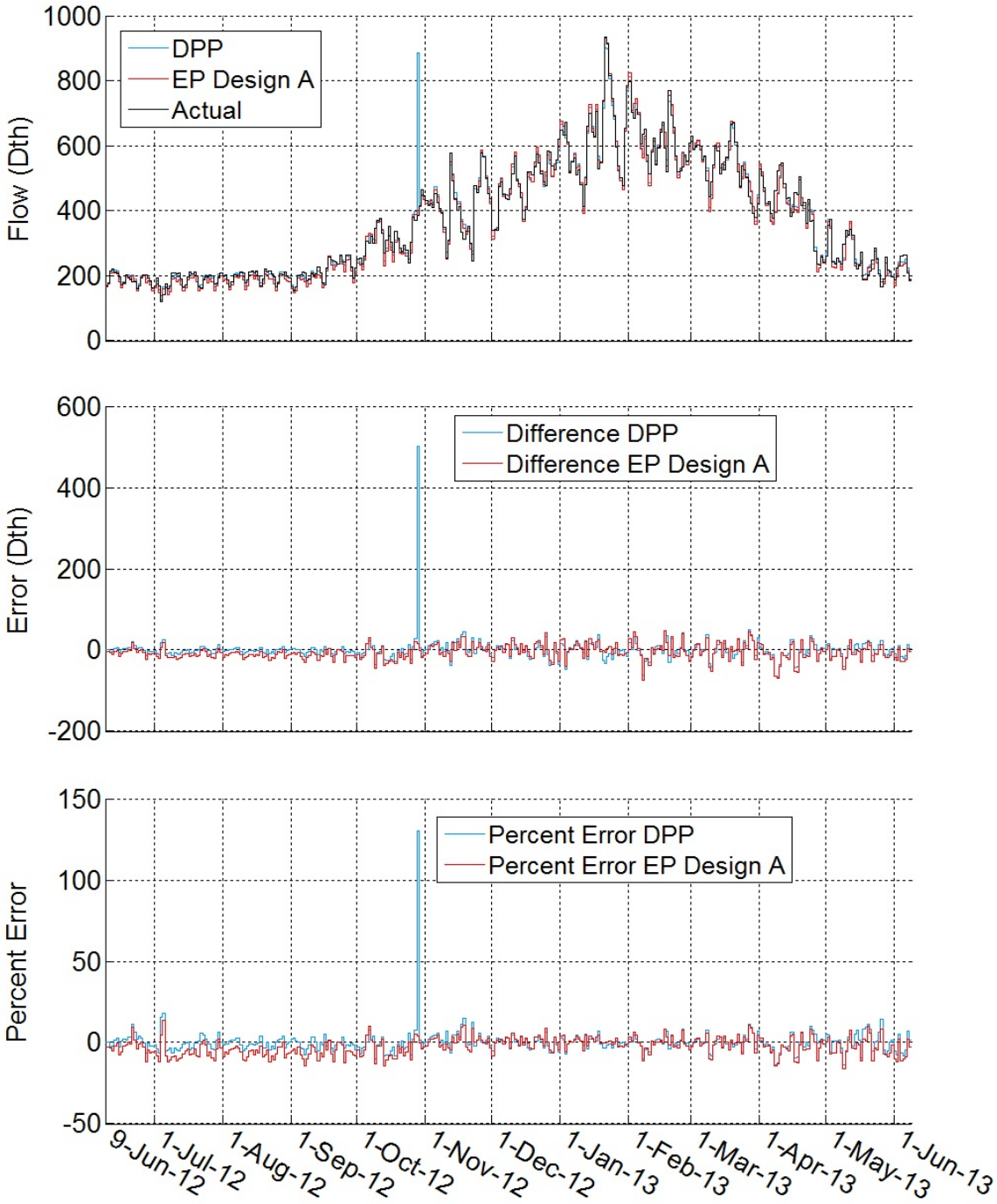

Figure 4.6: Time-series of operating area Bravo from June 2012 - June 2013

The graphs in Figure 4.7 compare the Dynamic Post Processor and ensemble model Design A errors decomposed by months. When looking at the months for the Standard Error values in the top graph, ensemble model Design A produced results that were a little worse than the Dynamic Post Processor. Ensemble model Design A performed better in November and January, while the Dynamic Post Processor performed better in almost every other month except for a few where the results were almost identical. These same observations hold true for the RMSE values, except that ensemble model Design A was greatly out–performed during the summer months.

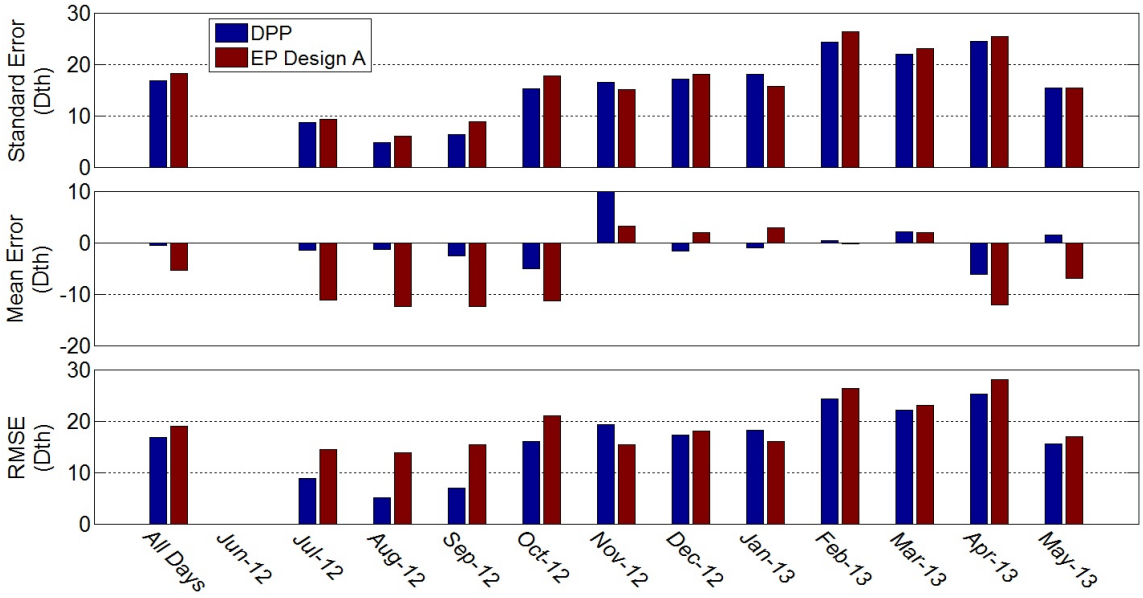

Figure 4.7: Operating area Bravo error decomposed by months

In Figure 4.8, the three graphs show the errors of both the Dynamic Post Processor and ensemble model Design A for all of the days in the testing set, as well

as the 10 different types of unusual days explained in Table 4.2. For the Standard Error values in the first graph, ensemble model Design A produced estimates that were just as good as or better than the Dynamic Post Processor on the "Coldest" days as well as on the days that were "Colder than Normal" and "Colder Today than Yesterday."

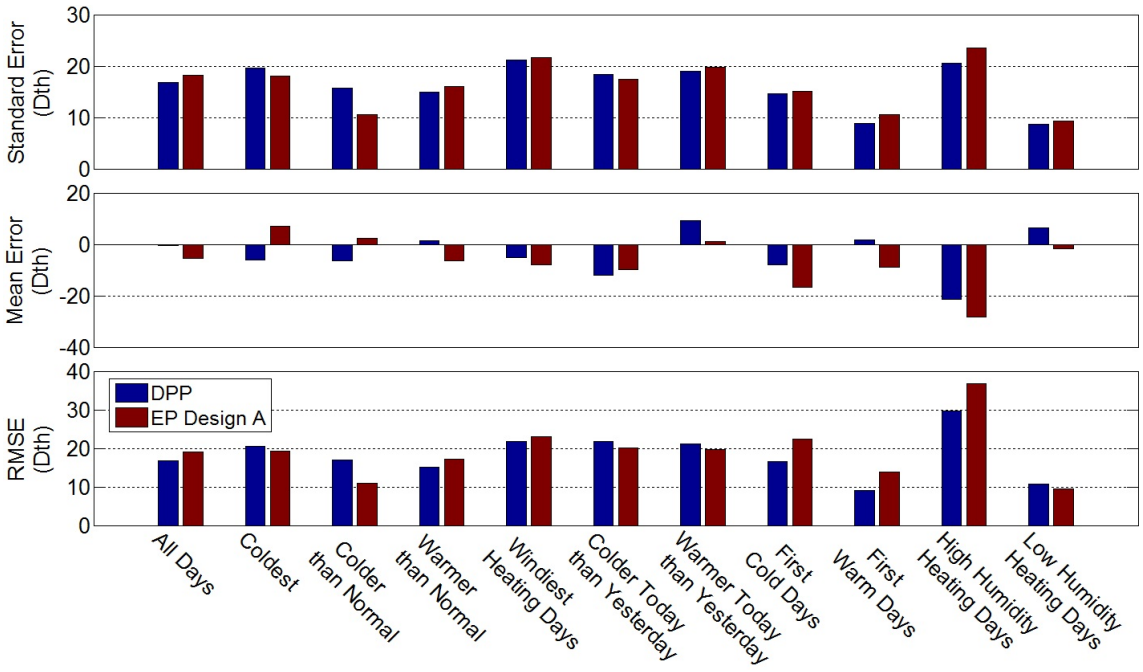

Figure 4.8: Operating area Bravo error decomposed by type of days

Unlike the results that are presented in Section 4.4, this comparison could go either way when looking only at the calculated errors. The Dynamic Post Processor performed better than ensemble model Design A overall, but ensemble model Design A performed better in some months and some types of days, specifically the "Coldest" days, where GasDay is expected to perform the best. However, the estimates are not accurate enough to warrant replacing the current Dynamic Post

Processor. This can also be shown by computing the T-statistic for the testing year of data. The computed T-value for ensemble model Design A's forecasts for "All Days" was not found to be statistically significant at conventional levels (5%).

4.6 Operating Area Charlie

A third operating area was chosen for comparing how well ensemble model Design A preformed compared to the Dynamic Post Processor because this operating area has typically had larger percent errors in the summer, instead of during the winter months. This operating area provides yet another type of area to show how robust the evolutionary programming engine is at producing ensemble models that are able to perform well in many different situations. The forecasts for ensemble model Design A were generated after the evolutionary programming engine ran for 100 generations for 800 different population members and took about three and a half days on a 20-core cluster to be determined.

In Figure 4.9, there are three time–series plots. The top plot shows the forecasts from the Dynamic Post Processor, ensemble model Design A, and the flow values from the Local Distribution Company's database. The bottom graph shows the percent error between both approaches; the percent errors are actually smaller during the winter months compared to the summer months, despite this utility being located in the northern part of the United States of America.

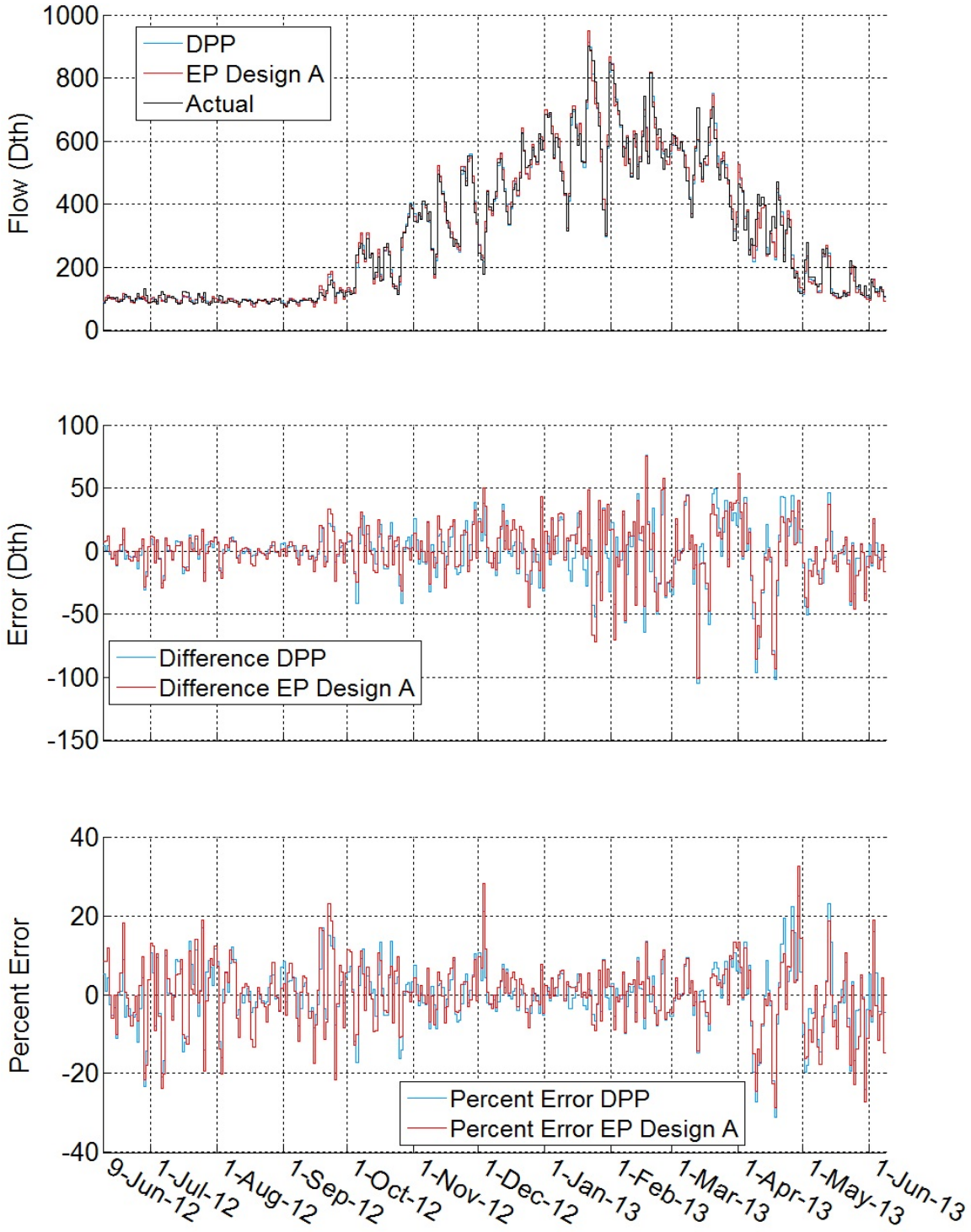

Figure 4.9: Time-series of operating area Charlie from June 2012 - June 2013

Figure 4.10 displays the errors for both the Dynamic Post Processor and ensemble model Design A decomposed by months. If we look at the Standard Error values (top graph), the errors appear almost identical. When comparing the individual months, there are months that the Dynamic Post Processor performs better, but there are also months that ensemble model Design A performs better. When looking at the RMSE values in the bottom graphs, we can see that the Dynamic Post Processor performed better in September and January, while ensemble model Design A performed better in March and April. The rest of the months were very similar in performance between the two models.

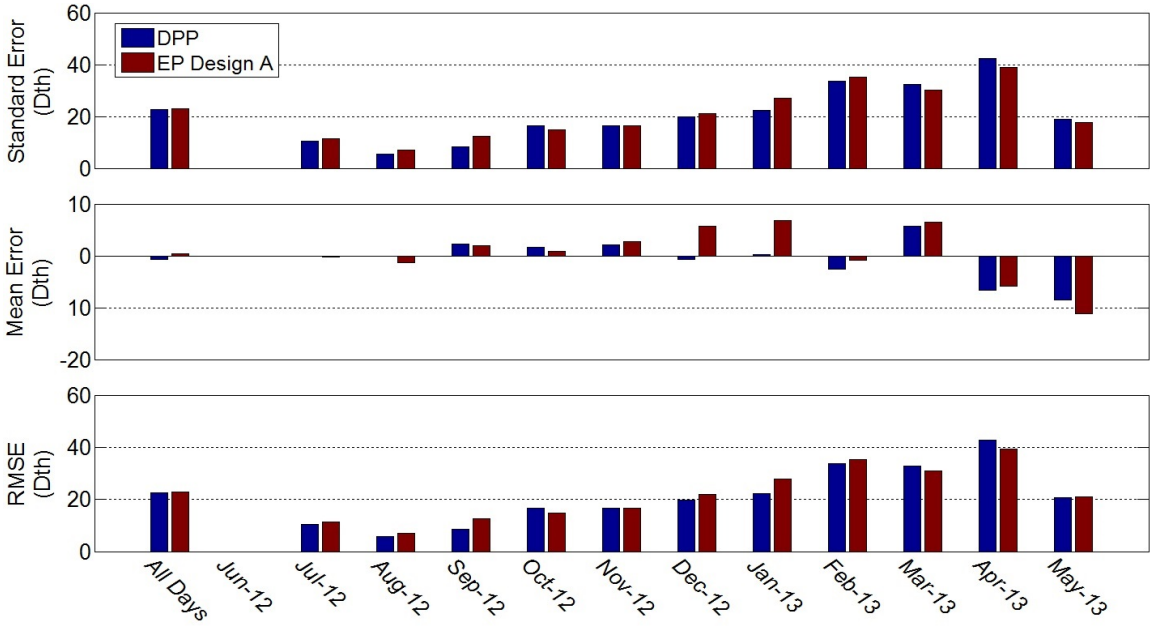

Figure 4.10: Operating area Charlie error decomposed by months

The third group of graphs presented for this operating area are shown in Figure 4.11 and shows the errors for both the Dynamic Post Processor and ensemble

model Design A for the unusual days specified in Table 4.2. As in the other operating areas, there are types of days where the Dynamic Post Processor performed better, but there are also types of days where ensemble model Design A performed better. In the third graph, the Dynamic Post Processor performs better on the "Coldest" days, while ensemble model Design A performed better on days that were "Colder than Normal," "Colder Today than Yesterday," and "Warmer Today than Yesterday."

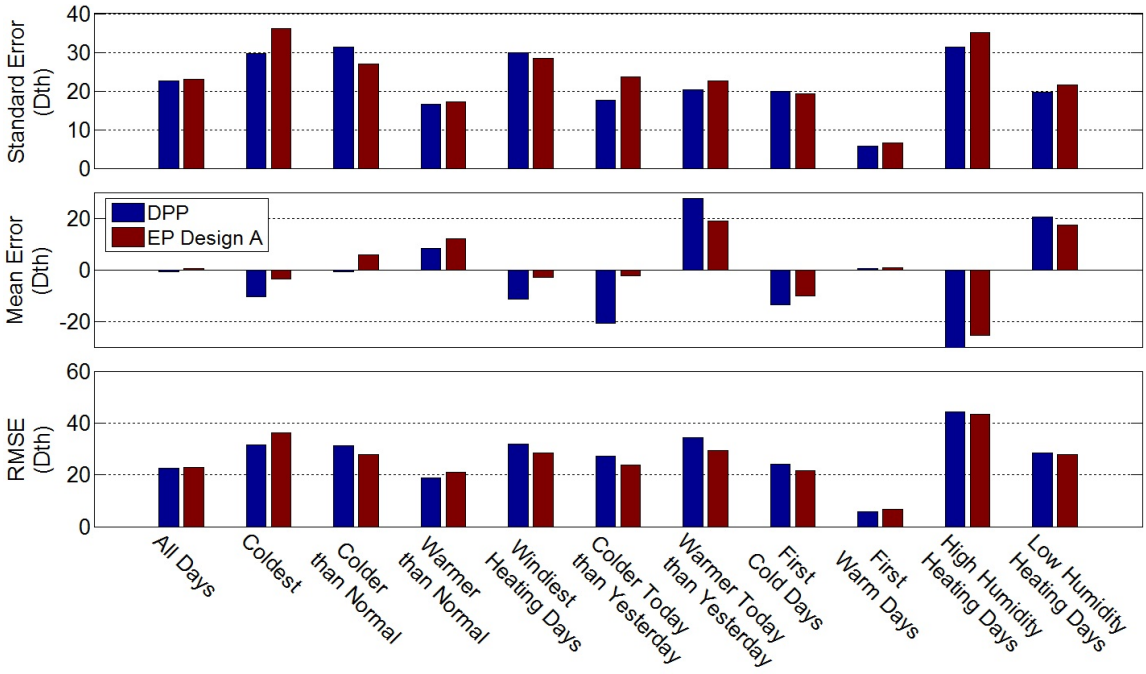

Figure 4.11: Operating area Charlie error decomposed by type of days

For this operating area, ensemble model Design A was able to produce forecasts that were more accurate than the Dynamic Post Processor, but when a T-test was performed on the results, the results were not found to be statistically significant at conventional levels (5%). This significance level, along with the

amount of computer time needed, makes ensemble model Design A the poorer choice of the two models to send to companies that license GasDay.

The fact that forecasts could be produced for three distinct operating areas that were either as good as or better than the Dynamic Post Processor suggests that this approach could be used on all 170 operating areas for which GasDay forecasts natural gas demand. To have evolutionary programming replace the current Dynamic Post Processor, testing would have to be done on all of the operating areas GasDay works with. However, by randomly picking three different operating areas that all had different characteristics, it shows that there is a chance that evolutionary programming could replace the Dynamic Post Processor, but additional testing is needed.

4.7 More Reasonable Results

The results presented in the rest of this chapter represent not the best results in terms of **accurate** forecasts, but in terms of **explainable** forecasts. Figure 4.12 shows the forecast flow from all 10 of the if–statements bodies regardless if the if–statement is true or false, the weighted linear combination of the four base component models and the overall estimate produced by ensemble model Design A for operating area Alpha. The weighted linear combination of the four component models is dominating the overall estimate, while some of the if–statement bodies

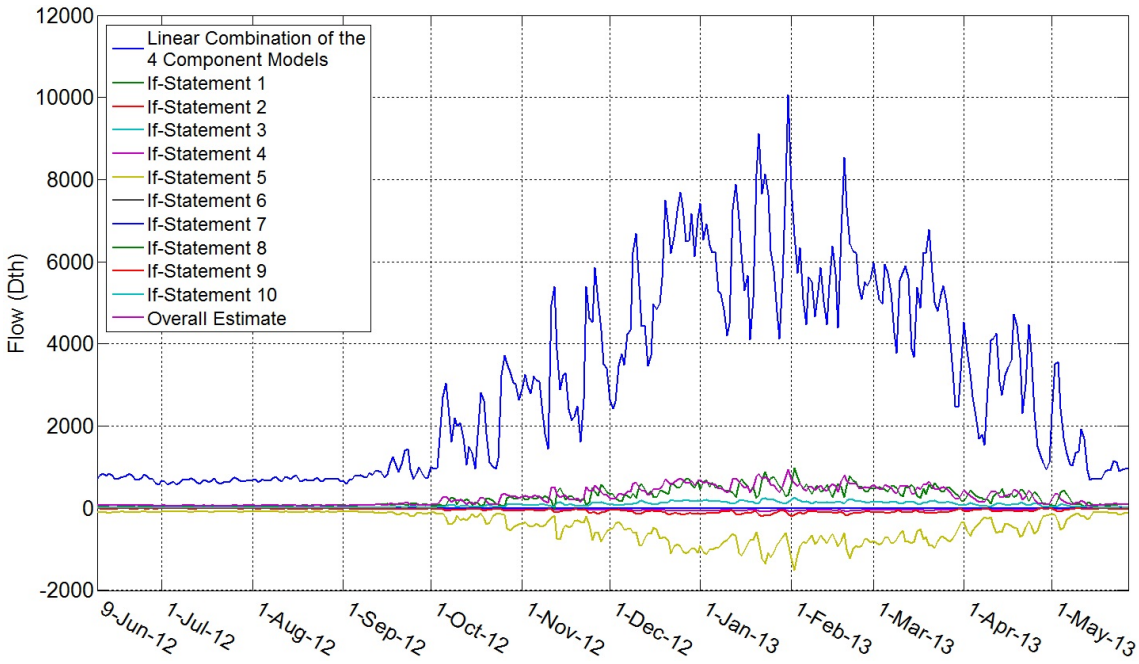

Figure 4.12: The individual if–statement estimates over the testing data from ensemble model Design A for operating area Alpha

produce negative results. The combination of these two facts makes explaining the actual ensemble model extremely difficult, because normally an ensemble model wants all of the individual components to be good estimates in their own right [46].

To address this concern, changes were made in the evolutionary programming ensemble model design to add the average of the if–statements to the weighted linear combination of the four base component models. Changes were also made to insure that all of the coefficients were positive. This change also drove the component estimates closer together. A visual representation of this is shown in Figure 4.13 and is further referred to as ensemble model Design D. This change also made sure that if none of the if–statements were true, an estimate would still be

produced that is reasonable, unlike what is possible based on the design discussed in the three operating-areas in the beginning of this chapter and shown in Figure 4.12 for operating area Alpha. The results produced by this change were not as accurate as previously discussed, so there is a trade-off between reasonability and results, which might benefit from further study. For the comparison, operating area Alpha was re-evaluated using this change in the code and compared to the results from the Dynamic Post Processor.

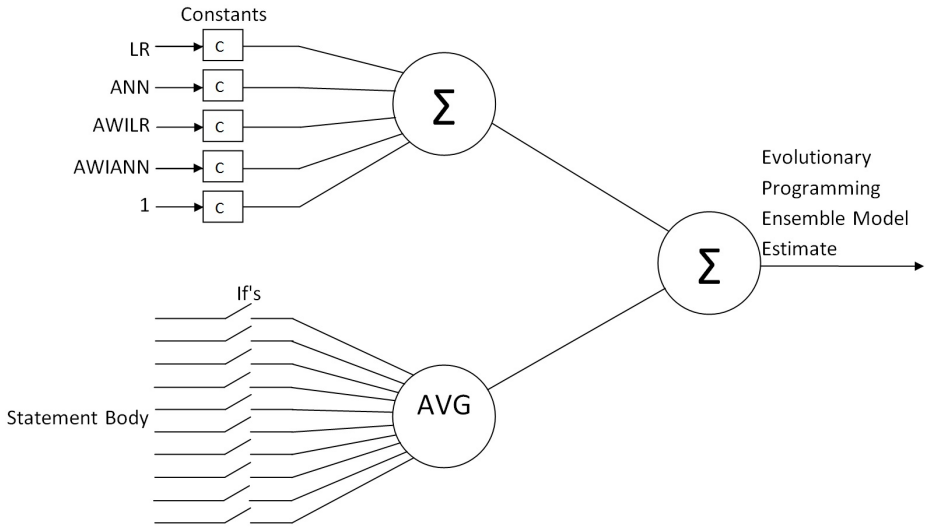

Figure 4.13: Evolutionary programming ensemble model Design D adding the average of the if-statements to the weighted linear combination of the 4 component models

In Figure 4.14, there are three time–series plots. The top plot shows the forecasts from the Dynamic Post Processor, ensemble model Design D and the flow values from the database. The bottom graph shows the percent error between both approaches; the percent errors are close during the middle of winter, but there is a huge difference in June, July, and August.

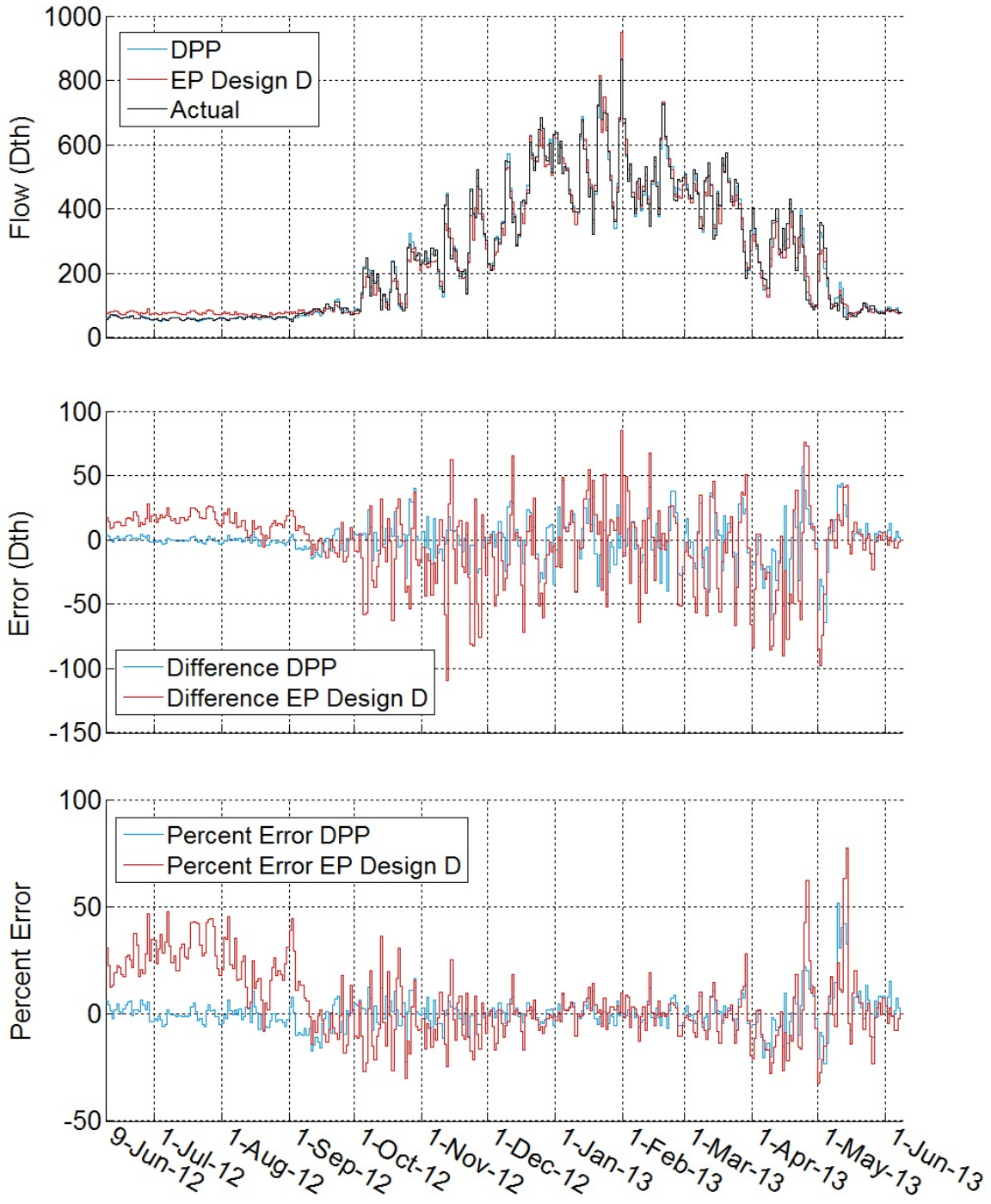

Figure 4.14: Time-series of operating area Alpha from June 2012 - June 2013

Figure 4.15 displays the errors for both the Dynamic Post Processor and ensemble model Design D decomposed by months. If we look at the Standard Error values, the errors for the Dynamic Post Processor are lower for every month, with November having the biggest difference between the Dynamic Post Processor and ensemble model Design D. The RMSE values in the bottom graph tell the same story. The Dynamic Post Processor performed better in every month. When we look at the Mean Error values in the middle graph, overall, ensemble model Design D did not perform much worse than the current Dynamic Post Processor, but the individual months tell a different story. There are months where ensemble model Design D had values that were better than the Dynamic Post Processor, September and February, but overall, it had Mean Errors that were larger than the Dynamic Post Processor.

The third group of graphs presented for this operating area are shown in Figure 4.16. This figure shows the errors for both the Dynamic Post Processor and ensemble model Design D for the unusual days specified in Table 4.2. Again, the Dynamic Post Processor performed better overall and on almost every type of unusual day except for "Low Humidity Heating Days."

As stated previously, the accuracy of this approach is not as good when compared with ensemble model Design A that was discussed in the beginning of this chapter, but the reasonability of the weighted linear combination of the four

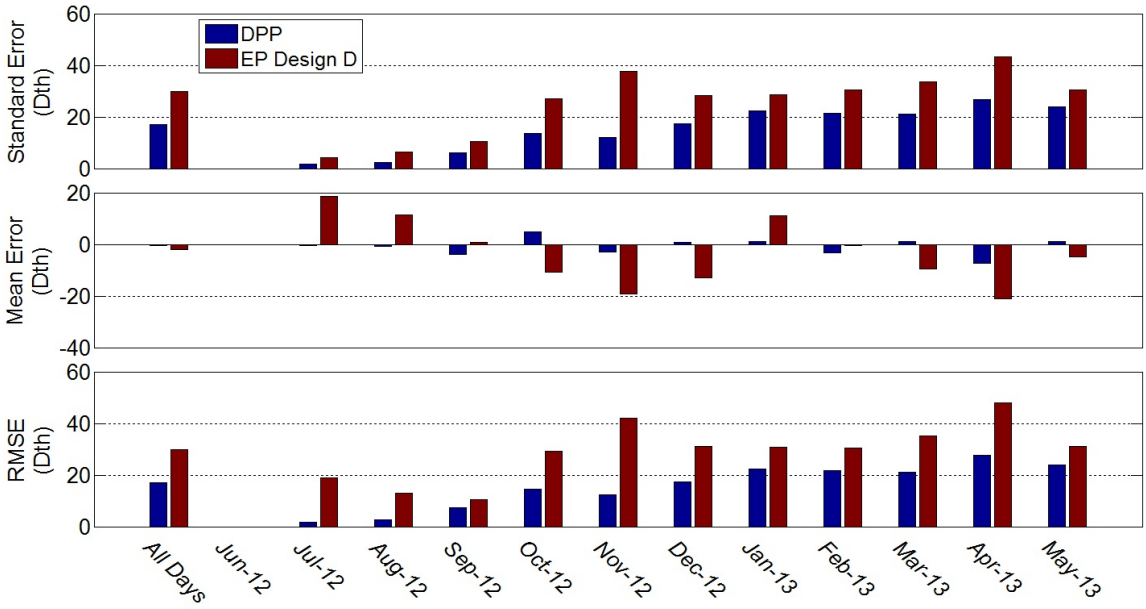

Figure 4.15: Operating area Alpha error decomposed by months

component models and 10 if–statements is higher. The results were not as accurate because by excluding part of the search space of the evolutionary programming engine, it excluded the area that was the most accurate, this can be further explained by Koza *et al.* They discuss a ratio between the knowledge the evolutionary program determines on its own divided by the knowledge that is supplied by the user [21]. By restricting the search space, we are adjusting this ratio away from pure evolutionary programming.

The increase in reasonability can be seen in Figure 4.17, where the estimate for the weighted linear combination of the four component models is very close to the overall estimate generated by ensemble model Design D with only small changes made based on the average of the if–statements that were true. All but a couple of

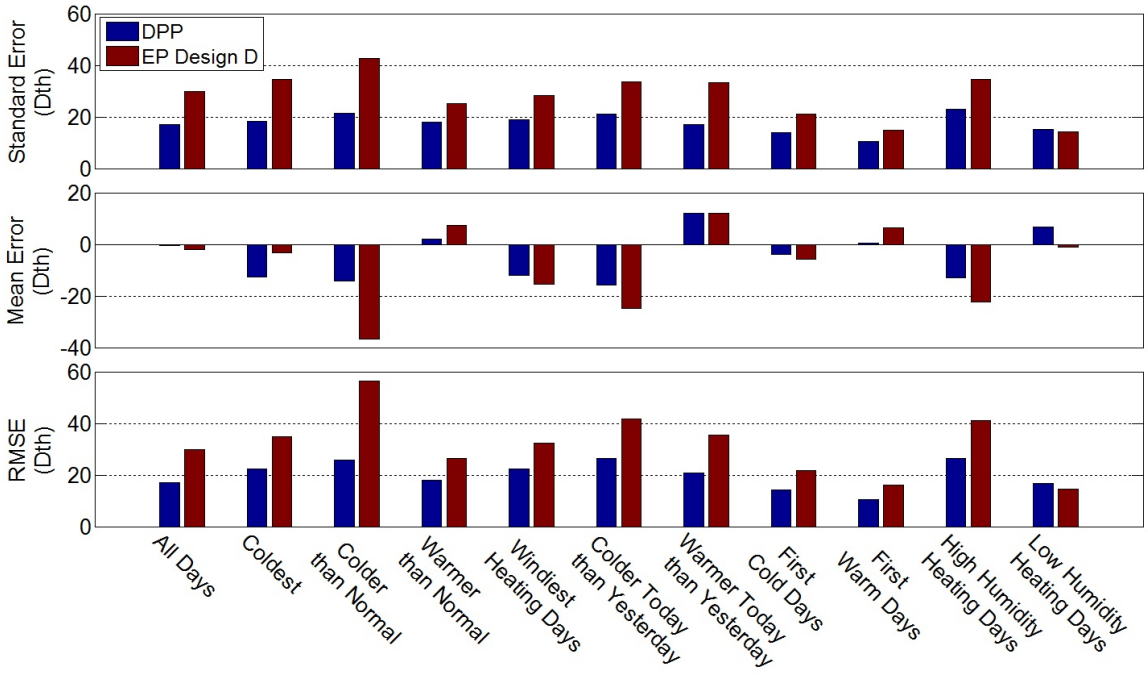

Figure 4.16: Operating area Alpha error decomposed by type of days

the if–statements are grouped together, with one of them, If–Statement 8, being a very good estimate in its own right. Most of the other if–statements are grouped together and their forecasts are quite a bit lower than the overall estimate generated by ensemble model Design D. This makes ensemble model Design D undesirable according to [46], but this design was the closest any of the experimented designs could come to quality forecasts across all models.

Figure 4.17 graphs the estimates for every day in the testing set regardless if the if–statement contributed to the overall forecast value or not. In Figure 4.18, the time–series chart shows a green dot every time that the conditional part of the if–statement is true and a red dot every time the conditional part of the

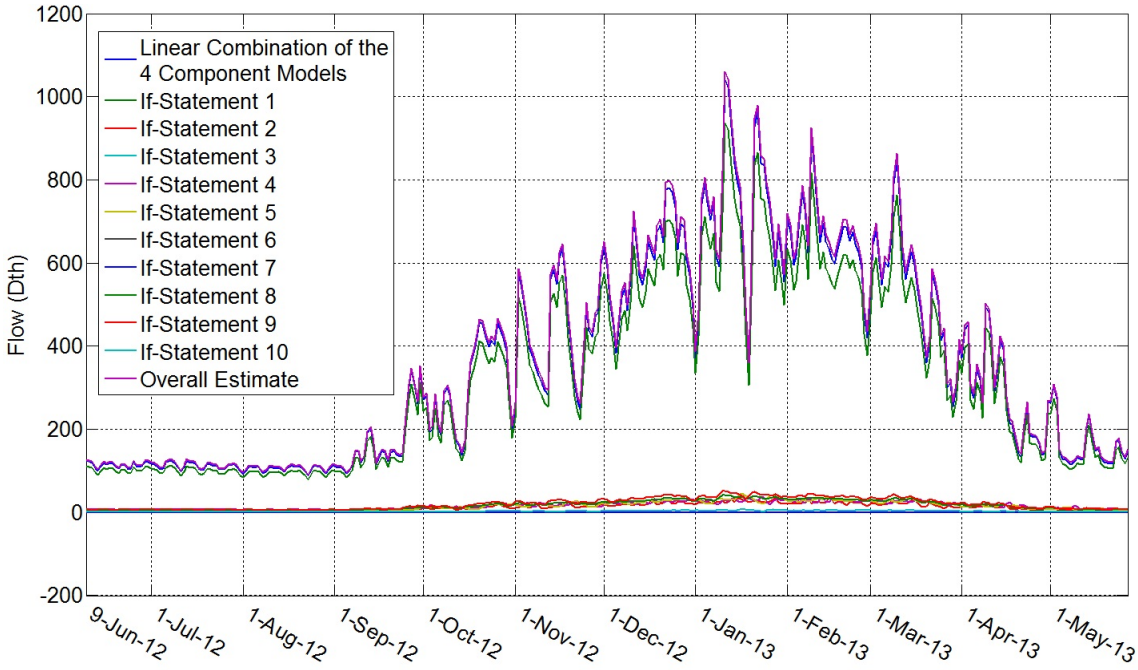

Figure 4.17: The individual if–statement estimates over the testing data from the design that averaged all 10 if–statements and added the weighted linear combination of the four base component models

if–statement is false. There are only two if–statements that switch for the entire year, If–Statements 1 and 2. Therefore, all 10 of the if–statements along with the weighted linear combination of the four component models are true 4% of the time, while 9 of the if–statements and the weighted linear combination of the four component models are true 96% of the time. Khan *et al.*, did research related to the topology of an artificial neural network using a neuro-evolutionary technique and discovered that the best performing designs statically used 5% – 10% of the input nodes that were initially available [19]. This is about the same results that ensemble model Design D demonstrated where the if-statements were switched about 4% of the time. The work done by Khan *et al.* is not a direct link to this specific case but

there is a parallel between the two bodies of work. This parallel is a relationship between what is expected and what the evolutionary program determines is the optimal solution. The if–statements should allow the ensemble model to make adjustments to the overall forecast while the addition of nodes in the artificial neural network should allow more complex relationships to be determined. Since the actual usage of the predetermined solutions was minimal, it shows that the evolutionary program will make any changes required to produce the lowest cost function, even if it uses a simpler design.

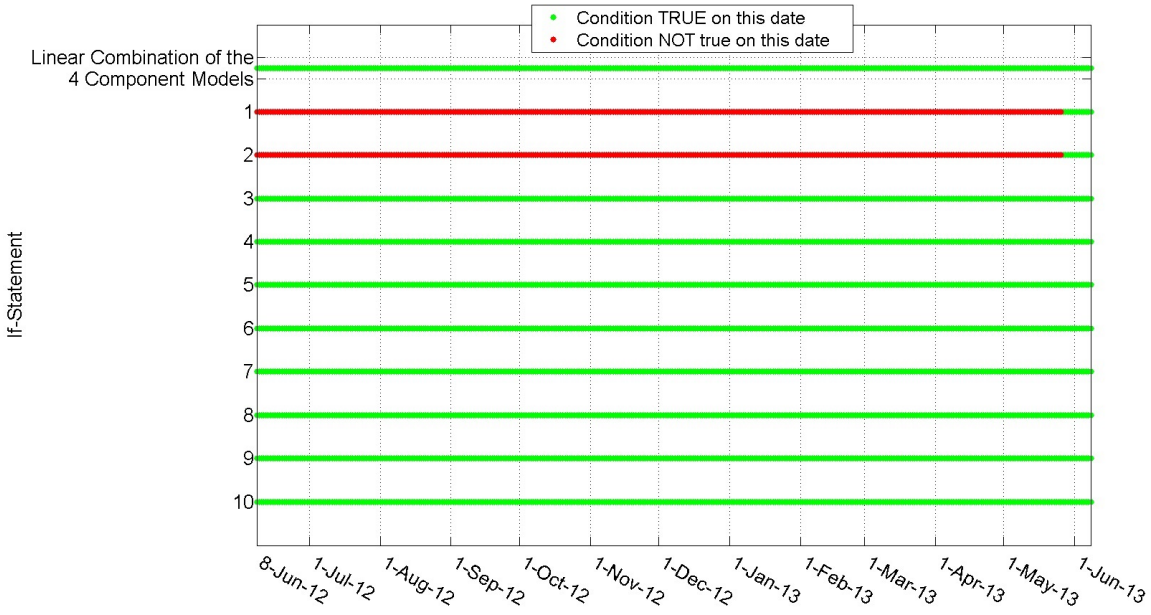

Figure 4.18: The individual if–statement evaluations for ensemble model Design D for every day in the testing set

The conditional part of If–Statements 1 and 2 are shown in Listing 4.1.

IF $\text{LR}_{6 \text{ days ago}} < C \times$ Recently Tuned Error, and

IF $\text{ANN}_{3 \text{ days ago}} < C \times$ Recently Tuned Error.

Listing 4.1: If–Statements 1 and 2 from ensemble model Design D

The if–statements were false most of the time because the estimates from the base component models were greater than the scaled Recently Tuned Error. For a description of the Recently Tuned Error variable, refer to Section 3.2.1. This held true until the beginning of summer when the base component models were estimating lower gas consumption than the scaled Recently Tuned Error, which grew over the winter months. If ensemble model Design D would have continued to be evaluated on data, eventually the if–statements would turn off again as the Recently Tuned Error would become smaller than the base component estimates.

While changes from ensemble model Design A were able to be made to increase the reasonability of the results, work still needs to be done to create an evolutionary programming ensemble model design that might eventually replace the Dynamic Post Processor. This work includes making sure that a different number of if-statements are evaluating true on different type of days and have all of the individual components producing a reasonable estimate.

4.8 Conclusion

This chapter presented results using ensemble model Design A, the most accurate during testing, and the results from ensemble model Design D, the most reasonable during testing. The results in this chapter were generated for Day 0, but the process can be developed for the other seven days that GasDay forecasts. To determine how well the evolutionary programming ensemble model design performs for the other seven days, the same types of graphs would be generated and analyzed to see the overall results from the evolutionary programming engine for the eight different time horizons.

The next chapter will also focus on different ensemble model designs that were used while working on this thesis. The designs did not perform as accurately or produce forecasts that were as reasonable as the designs that were discussed in this chapter. They are included to provide additional assistance to others who would like to expand and build upon the work that is presented in this thesis so time is not wasted copying work that was already completed.

CHAPTER 5

ADDITIONAL EVOLUTIONARY PROGRAMMING ENSEMBLE MODEL DESIGNS

This chapter will discuss additional ensemble model designs that were evaluated throughout this thesis. The chapter is broken up into sections based on the design of the evolutionary programming ensemble model. This chapter also gives a description of the time and computing power that was needed to complete the work for this thesis.

5.1 Ensemble Model Design B (average) and C (sum)

Table 4.1 presents the RMSE values from the three best performing ensemble model designs we considered in Chapter 4. Diagrams of the other two evolutionary programming ensemble model designs that were not used to produce the results in the first half of Chapter 4 are shown in Figures 5.1 and 5.2.

5.1.1 Ensemble Model Design B

The second ensemble model design that was considered is a linear combination of the four current GasDay models. The coefficients for the four base component

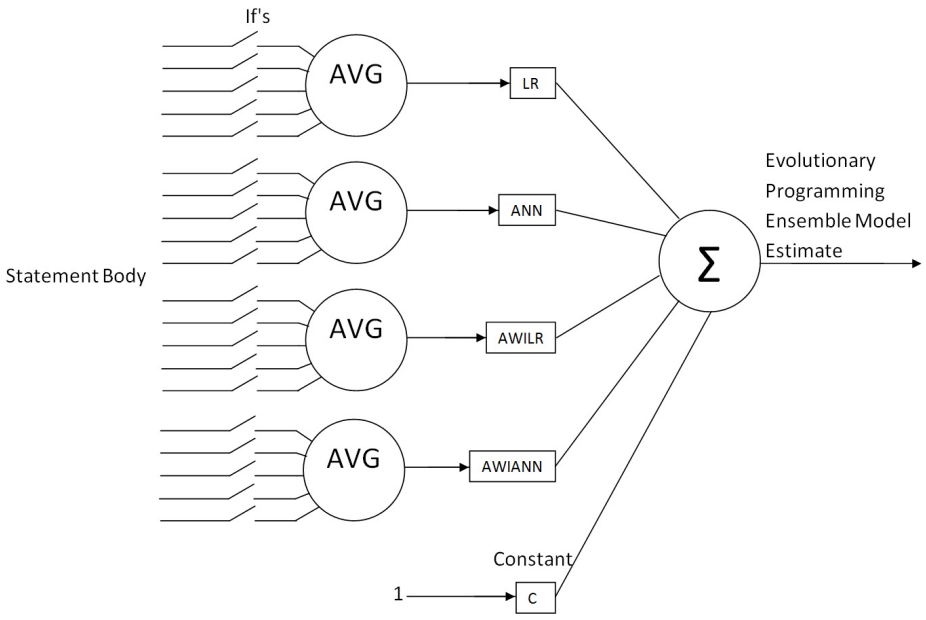

Figure 5.1: Evolutionary programming ensemble model Design B using averages

models are determined by the **average** of up to five if–statement bodies, as suggested by Figure 5.1.

As shown in Table 4.1, ensemble model Design B did not perform very well. There were only five if–statements assigned to each coefficient, and those five if–statements had to account for a wide range of inputs and produce a wide range of values. Even though there were only five if–statements assigned to each model coefficient, there were still 20 if–statements for the entire population member, and `fminunc` had to determine 81 constants for each member of the population.

5.1.2 Ensemble Model Design C

The third ensemble model design that was considered was identical to ensemble model Design B, except that the coefficients of the four base component models were a **summation** of all of the executed if–statement bodies, instead of the **average**. The summation of the if–statement bodies did make for a more accurate design as discussed earlier but the design is less resistant to bad data. It is less resistant to bad data because the average limited the error generated by one if–statements by the number of if–statements that were true. The summation of the if–statements allowed each statement to not have to generate an estimate of the ideal coefficient value, but if an if–statement was not very good at producing an estimate of the coefficient, the evolutionary programming engine could adjust the if–statement constants to provide an offset to the other four statements. As with ensemble model Design B that was tried, ensemble model Design C resulted in each population member having 20 if–statements and a total of 81 constants. All of the constants were determined by `fminunc` to minimize the error on the training data. A diagram of ensemble model Design C is shown in Figure 5.2.

The next section of this chapter covers a variation of ensemble model Design D that was presented in Section 4.7. Though this design was not tested thoroughly, it was run at the end of the thesis to see if a better design could be found based on increased knowledge and using a model that was known to be accurate.

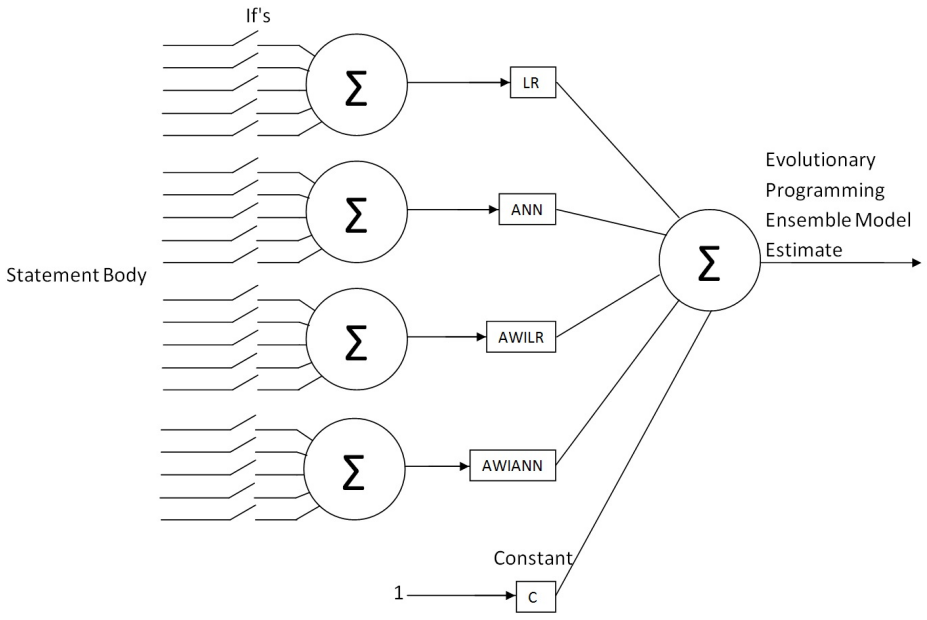

Figure 5.2: Evolutionary programming ensemble model Design C using summations

5.2 Ensemble Model Design E

A variation of the design that was discussed in Section 4.7 was to remove the linear combination of the four base component models, and replace it with the estimate that was generated by the Dynamic Post Processor. Ensemble model Design E should have produced forecasts no worse than the forecasts that were generated by the Dynamic Post Processor because there was more freedom to adjust the estimates in addition to what the Dynamic Post Processor already does. A diagram showing this design is in Figure 5.3.

This ensemble model design was generated by the evolutionary programming engine using a population size of 800 members for 100 generations for operating

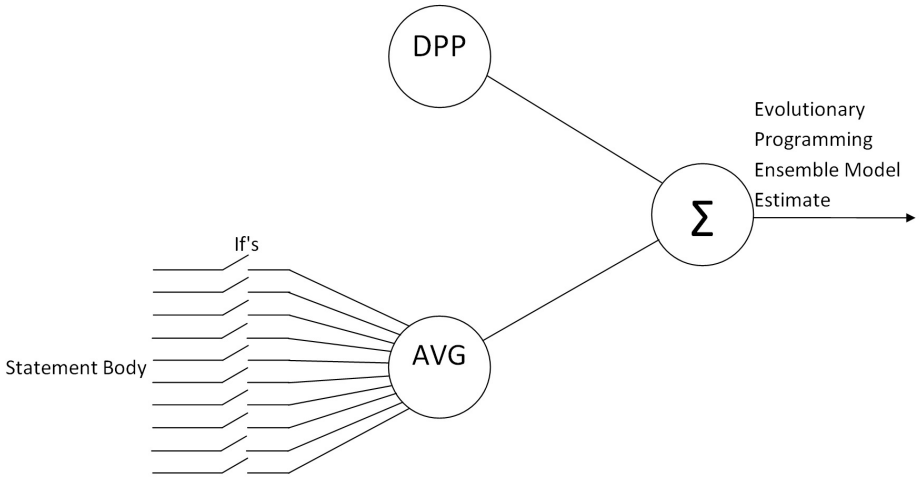

Figure 5.3: Evolutionary programming ensemble model Design E using the Dynamic Post Processor

area Alpha, and Figure 5.4 shows the error of ensemble model Design E decomposed by months. The design was able to perform better than just the Dynamic Post Processor across "All Days," as the if–statements allowed another mechanism to further model the errors for this specific operating area. Ensemble model Design E performed worse than the current Dynamic Post Processor during the first few months of comparison, but once the weather turned colder, ensemble model Design E performed the same or better for every month.

5.3 CPU Time

As mentioned previously in Section 2.4 and throughout this thesis, all of the results presented in this thesis were obtained by allowing the evolutionary programming engine to run for 100 generations on a population size of 800 members, and the

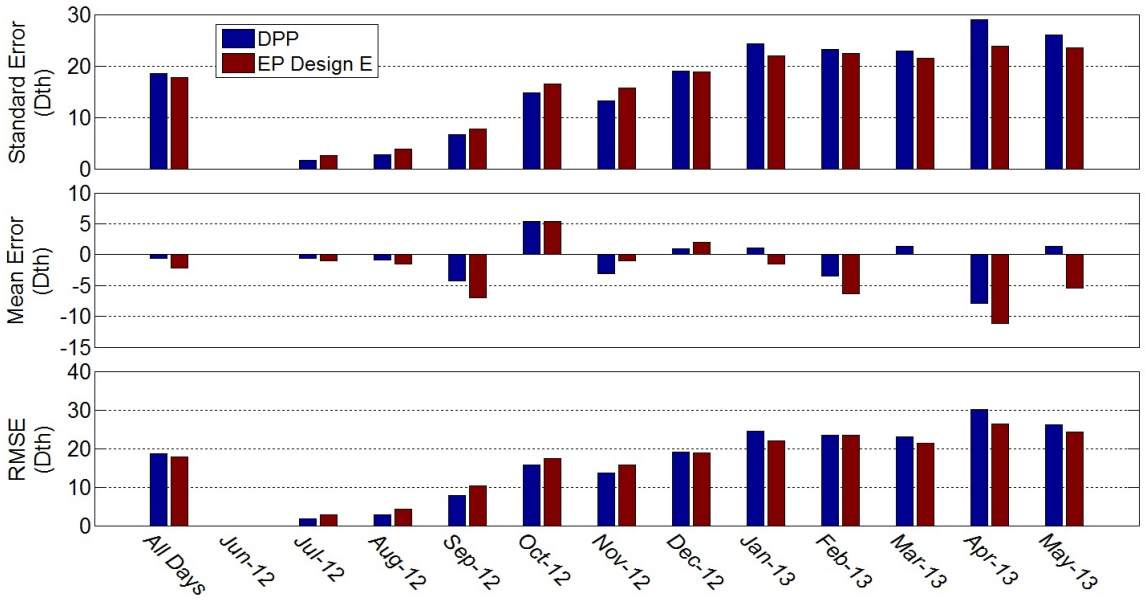

Figure 5.4: Operating area Alpha error decomposed by month for ensemble model Design E

constants were allowed to be optimized up to 1000 times per generation using `fminunc`. This size of a run took a lot of CPU time to complete. On a two–core machine, it took 6.5–7.5 hours per generation. This amount of time is too long for multiple runs. This is why the GasDay 20–core cluster was used. The head node of the cluster was running Windows XP, while all 20 of the worker nodes were running Windows 7. 16 of the workers were Intel Xeon processors running at 2.40 GHz and four of the workers were Intel i7 processors running at 3.40GHz. Using the cluster, a single generation of 800 members could be completed in about 55 minutes.

For this thesis, five different runs were completed for a total of 500 generations. The code for this thesis was structured so that after every generation, the population members and constants were saved to a *.mat file which allowed the

different runs to be completed on multiple computers simply by moving the *.mat files from one machine to another. This also provided a backup in case the cluster or MATLAB crashed. This is also known as checkpointing [4]. This implementation allowed normally idle computers to be used until the cluster was finished on the previously assigned run and then move the *.mat files and start at the generation at which the other computer left off. This saved some time because multiple runs were being completed at the same time, even though the throughput on the individual computers was 6–7 times slower than the cluster.

5.4 Conclusion

This chapter discussed the evolutionary programming ensemble model designs that produced results that were not as accurate as ensemble model Design A or as reasonable as ensemble model Design D that were discussed in Chapter 4. This chapter also discussed the computer power that was needed to generate the results presented in this chapter and in Chapter 4. The next chapter will conclude this thesis with overall observations of the work presented, and discuss how this thesis can be used as a foundation for future work in the area of natural gas forecasting using evolutionary programming.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

The goal of this thesis was to create individual ensemble models for different operating areas using evolutionary programming that would produce accurate gas demand forecasts for time horizon 0. By using work presented by Koza *et al.* [21] and Roebber [30], an evolutionary programming engine and an evolutionary programming ensemble model design were created that were able to produce results that were more accurate than the current Dynamic Post Processor. These results however, were not statistically more accurate than the current Dynamic Post Processor as shown in Chapter 4. The produced results also were not reasonable when we looked past the raw error values, and the ensemble models took days of computing power on a cluster to be determined.

This work also expanded the technique of evolutionary programming into a previously unexplored area of natural gas forecasting. This thesis also demonstrated how an evolutionary program responds to using inputs that are already good solutions to the problem that is trying to be solved. As discussed in Section 4.7, this thesis also worked to apply ensemble model forecasting theory to evolutionary

programming by trying to make all of the individual if–statements reasonable estimators in their own right [46].

Evolutionary Programming was first introduced in Chapter 2; its extension to natural gas demand forecasting was presented in Chapter 3. Chapter 3 gave the details of how the evolutionary programming engine functions as well as the design of the evolutionary programming ensemble model that was used to produce the results in the first part of Chapter 4. The second half of Chapter 4 along with Chapter 5 showed ensemble model designs that were not able to produce results that were as accurate as the Dynamic Post Processor. The results in the first half of Chapter 4 showed through both Standard Error (Std Error) as well as Root Mean Square Error (RMSE) that ensemble model Design A was able to forecast daily natural gas demand through the different if–statements that were created and evolved by the genetic algorithm. However, these results were not statistically significant, required too much computer power to be done for all approximately 170 operating areas where GasDay forecasts natural gas demand, and did not have individually reasonable component model estimates.

6.2 Future Research

Although the work presented in this thesis was able to produce forecasts that were more accurate than the Dynamic Post Processor based on pure error calculations for

three operating areas that each had a unique characteristic, this work still represents the first approach of using evolutionary programming to forecast natural gas demand; there are many improvements that can be considered. We list several possible future research areas below.

A full exploration of the potential of different ensemble model designs and evolutionary programming requires more computing power than was available for this work. We would like to explore more generations and larger populations.

As mentioned in Chapter 4, ensemble model Design A was chosen because it performed the best of ensemble model Designs A, B, and C in an experiment using a small population size and a small number of generations. Additional testing was done with other designs, which may yet offer further accuracy and speed performance improvements. These designs may also build on the explainable results that were presented in Section 4.7, whose overall accuracy, individual model forecasts, and activity of the if–statements were not as good as desired. One approach to increase the number of if–statements that are actively switching is to add to the cost function a term that reflects the number of if–statement combinations that are present during testing and validation. This is similar to the regularization technique discussed in Section 2.2, where the squared values of the coefficients are added to the cost function to help reduce the magnitude of the coefficients.

Additional research can be done to determine the optimal number of years for training of the constants and for validation. In Chapters 3, 4, and 5, it was mentioned that three years of data was used for training the constants, and one year was used for validation. Many of the Local Distribution Company's databases have data going back over a decade. Should more years of data be used to help increase the accuracy of the evolutionary programming ensemble model, as it would have been exposed to more years of data for both training as well as validation? A variant might be to use bootstrapping [20] to determine the values of the constants. If the constants are close, the model is more likely to move to the next generation, compared to a model whose constants are vastly different from one training data set to another.

Work can also be done to incorporate surrogate data. Surrogate data transforms data from one operating area to another by changing the characteristics of the data to match the current operating area [3]. This allows a richer set of data to be used. This is especially true for unusual days. Normally, a database only has a few unusual days, so increasing the number of unusual days on which the ensemble models can be trained increases the accuracy of the ensemble models in the future since it will have been trained on more of the days.

As mentioned in Section 3.2.1, specific inputs were chosen because they have been known to be important for forecasting natural gas demand, but the Local

Distribution Company's databases have additional data sets that can be used. An additional possible input is the 30-year expected average temperature. This temperature is used for determining unusual day types, but is not used as an input to the ensemble model. Other possible inputs are the component model forecasts that were made for the specific day at different time horizons. If we are trying to forecast the demand for today, we could also use the base component estimates from the previous seven time-horizon forecasts for the specific day as inputs.

Throughout this thesis, work was done on both the type of if–statement bodies, as well as the total number of if–statements. In Chapter 5, it was discussed that only using five if–statements seemed insufficient, but is 10 the ideal number to use? Work can also be done to allow the evolutionary programming engine to determine the number of statements that are present in each ensemble model, in addition to determining the variables that are used and how they are combined. An example of this is in Section 5.2, where ensemble model Design D used the forecasts of the Dynamic Post Processor in place of the linear combination of the four base component models to produced forecasts that were more accurate than just the Dynamic Post Processor.

In conclusion, the goal of this thesis was to use the concept of evolutionary programming to develop an ensemble model that was able to produce accurate forecasts for different operating areas where GasDay forecasts demand. The results

presented in Chapter 4 showed that the chosen evolutionary programming ensemble model design did perform better than the current Dynamic Post Processor when looking at the error values. However, the amount of time required to produce the results and the reasonability of the individual if–statement forecasts were not enough to support the claim that an evolutionary programming approach is ready for prime time.

BIBLIOGRAPHY

[1] J. Ai-ping and H. Feng-wen, "Methods for Optimizing Weights of Wavelet Neural Network Based on Adaptive Annealing Genetic Algorithm," *Industrial Engineering and Engineering Management*, pp. 1744–1748, 2009.

[2] ANGA.org, "Ratemaking for Energy Pipelines," `http://www.aga.org/our-issues/RatesRegulatoryIssues/Documents/Ratemaking%20for%20Energy%20Pipelines.pdf`, 2013, accessed: 4/2/2014.

[3] J. Armstrong, *Principles of Forecasting - A Handbook for Researchers and Practitioners*. Boston, MA: Kluwer Academic, 2001.

[4] M. Bouguerra, A. Gainaru, L. Gomez, F. Cappello, S. Matsuoka, and N. Maruyama, "Improving the Computing Efficiency of HPC Systems using a Combination of Proactive and Preventive Checkpointing," *IEEE 27th International Symposium on Parallel & Distributed Processing*, pp. 501–512, 2013.

[5] M. Butler and D. Kazakov, "The Effects of Variable Stationarity in a Financial Time-Series on Artificial Neural Networks," *Computational Intelligence for Financial Engineering and Economics*, pp. 1–8, 2011.

[6] A. Ceballos, "Natural Gas & Shale Gas Drilling [Hydraulic Fracturing]," `http://tnsuniversitycenter.com/2012spring/?p=760`, 2012, accessed: 6/19/2013.

[7] M. Chu-Carroll, "Science Blogs," `http://scienceblogs.com/goodmath/2006/11/02/the-c-is-efficient-language-fa/`, 2006, accessed: 1/15/2014.

[8] E. Contreras-Hernandez and J. Cedeno-Maldonado, "A Sequential Evolutionary Programming Approach to Profit-Based Unit Commitment," *Transmission & Distribution Conference and Exposition*, pp. 1–8, 2006.

[9] L. Dalton and E. Dougherty, "Bayesian Minimum Mean-Square Error Estimation for Classification Error Part I: Definition and the Bayesian MMSE Error Estimator for Discrete Classification," *IEEE Transactions*, vol. 59, no. 1, pp. 115–129, 2011.

[10] Dictionary.com, "BTU—define BTU at dictionary.com,"

http://dictionary.reference.com/browse/btu, 2013, accessed: 6/25/2013.

[11] DTEEnergy.com, "Natural Gas Processing, Delivery and Storage," http://www.dteenergy.com/residentialCustomers/productsPrograms/gas/gasDelivery.html, 2013, accessed: 6/19/2013.

[12] Eia.gov, "Industrial Sector Natural Gas Use Rising," http://www.eia.gov/-todayinenergy/detail.cfm?id=11771, 2013, accessed: 8/20/2013.

[13] L. J. Fogel, P. J. Angeline, and T. Back, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming (Complex Adaptive Systems)*. San Diego, CA: The MIT Press, October 1996.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston, MA: Addison-Wesley Publication, 1989.

[15] J. G. D. Gooijer and R. J. Hyndman, "25 Years of Time Series Forecasting," *International Journal of Forecasting*, vol. 22, pp. 443–473, 2006.

[16] P. Guo, X. Wang, and Y. Han, "The Enhanced Genetic Algorithms for the Optimization Design," *Biomedical Engineering and Informatics*, vol. 7, pp. 2990–2994, 2010.

[17] J. Heaton, *Introduction to Neural Networks with Java*. Chesterfield, MO: Heaton Research, 2005.

[18] Justjoules.org, "Just Joules – Dedicated to Unifying Our Energy Terminology," http://www.http://www.justjoules.org/conversions.html, 2007, accessed: 2/13/2014.

[19] G. Khan, S. Khan, and F. Ullah, "Short-Term Daily Peak Load Forecasting using Fast Learning Neural Network," *Intelligent Systems Design and Applications*, pp. 843–848, 2011.

[20] J. P. Kleijnen, A. J. Feelders, and R. C. Cheng, "Bootstrapping and Validation of Metamodels in Simulation," *Simulation Conference Proceedings*, vol. 1, pp. 701–706, 1998.

[21] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza,

Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Springer, March 2005.

[22] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, NJ: Prentice Hall, 1999.

[23] Mathworks.com, "Unconstrained Nonlinear Optimization Algorithms," `http://www.mathworks.com/help/optim/ug/unconstrained-nonlinear-optimization-algorithms.html#brnoxxo`, accessed: 11/22/2013.

[24] NaturalGas.org, "Background," `http://www.naturalgas.org/overview/background.asp`, 2013, accessed: 7/3/2013.

[25] ——, "The Transportation of Natural Gas," `http://www.naturalgas.org/naturalgas/transport.asp`, 2013, accessed: 7/3/2013.

[26] B. Otok, D. A. Lusia, Suhartono, R. Faulina, Sutikno, and H. Kuswanto, "Ensemble Method Based on ARIMA-FFNN for Climate Forecasting," *Statistics in Science, Business and Engineering*, pp. 10–12, 2012.

[27] X. Pan and J. Wu, "Bayesian Neural Network Ensemble Model Based on Partial Least Squares Regression and its Application in Rainfall Forecasting," *Computational Sciences and Optimization*, vol. 2, pp. 49–52, 2009.

[28] B. Pang, "The Impact of Additional Weather Inputs on Gas Load Forecasting," Master's thesis, Marquette University, Department of Electrical and Computer Engineering, Milwaukee, WI, August 2012.

[29] P. J. Roebber, "Seeking Consensus: A New Approach," *Monthly Weather Review*, vol. 138, pp. 4402–4415, 2010.

[30] ——, "Using Evolutionary Programming to Generate Skillful Extreme Value Probabilistic Forecasts," Presentation to GasDay, June 2013.

[31] R. Sathyanarayan, H. Birru, and K. Chellapilla, "Evolving Nonlinear Time-Series Models using Evolutionary Programming," *Evolutionary Computation*, vol. 1, pp. 243–250, 1999.

[32] A. Sheta and A. Mahmoud, "Forecasting using Genetic Programming," *System Theory, 2001. Proceedings of the 33rd Southeastern Symposium*, pp. 343–347, 2001.

[33] Y. Tai-shan, "An Improved Genetic Algorithm and Its Blending Application with Neural Network," *Intelligent Systems and Applications*, pp. 1–4, 2010.

[34] J. Taylor and R. Buizza, "Neural Network Load Forecasting with Weather Ensemble Predictions," *IEEE Transactions*, vol. 17, no. 3, pp. 626–632, 2002.

[35] TeXample.net, "Neural Network," `http://www.texample.net/tikz/examples/neural-network/`, accessed: 6/10/2013.

[36] Tribal Energy and Environmental Information, "Oil and Gas Resources and Their Uses," `http://teeic.anl.gov/er/oilgas/restech/uses/index.cfm`, 2013, accessed: 6/25/2013.

[37] U.S. Energy Information Administration (EIA), "How Much Natural Gas is Consumed (used) in the U.S.?" `http://www.eia.gov/tools/faqs/faq.cfm?id=50&t=8`, 2013, accessed: 6/17/2013.

[38] A. Vahidi, A. Stefanopoulou, and H. Peng, "Recursive Least Squares with Forgetting for Online Estimation of Vehicle Mass and Road Grade: Theory and Experiments," *Vehicle System Dynamics*, vol. 43, no. 1, pp. 31–55, 2005.

[39] V. Vapnik and C. Cortes, "Support - Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[40] S. R. Vitullo, "Disaggregating Time Series Data for Energy Consumption by Aggregate and Individual Customer," Ph.D. dissertation, Marquette University, Department of Electrical and Computer Engineering, Milwaukee, WI, March 2011.

[41] L. Wang and J. Wu, "Application of Hybrid RBF Neural Network Ensemble Model Based on Wavelet Support Vector Machine Regression in Rainfall Time Series Forecasting," *Computational Sciences and Optimization*, pp. 867–871, 2012.

[42] S. Wang, B. Meng, and H. Tian, "A Modeling Method Based on Wavelet Support Vector Machine," *Control and Decision Conference (CCDC)*, pp. 26 – 28, 2010.

[43] G. Wei, "Study on Genetic Algorithm and Evolutionary Programming," *Parallel Distributed and Grid Computing (PDGC)*, pp. 762–766, 2012.

[44] C. R. Wilkes, M. D. Koontz, and I. H. Billick, "Analysis of Sampling Strategies for Estimating Annual Average Indoor NO_2 Concentrations in Residences with Gas Ranges," *Journal of the Air and Waste Management Association*, vol. 46, no. 9, pp. 853 – 860, 1996.

[45] C. Woodford, "How Neural Networks Work - A Simple Introduction," `http://www.explainthatstuff.com/introduction-to-neural-networks.htm`, 2013, accessed: 6/10/2013.

[46] World Meteorological Organization, *Guidelines on Ensemble Prediction Systems and Forecasting*. Geneva, Switzerland, 2012.

[47] J. Wu, L. Huang, and X. Pan, "A Novel Bayesian Additive Regression Trees Ensemble Model Based on Linear Regression and Nonlinear Regression for Torrential Rain Forecasting," *Computational Science and Optimization*, vol. 2, pp. 466–470, 2010.

[48] K. Yang and C. Shahabi, "On the Stationarity of Multivariate Time Series for Correlation-Based Data Analysis," *Data Mining*, pp. 27–30, 2005.

[49] L. Zhang, W. Zhou, and L. Jiao, "Wavelet Support Vector Machine," *Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 34 – 39, 2004.

[50] F. Zheng and S. Zhong, "Time Series Forecasting using an Ensemble Model Incorporating ARIMA and ANN based on Combined Objectives," *Artificial Intelligence, Management Science and Electronic Commerce*, pp. 2671–2674, 2011.