Exploration of the Potential for Gene Expression Programming to Solve some Problems in Meteorology and Renewable Energy

by

Atoossa Bakhshaii Shahrbabaki

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES
(Atmospheric Science)

The University of British Columbia
(Vancouver)

April 2012

Abstract

This dissertation describes research to enhance hydrometeorological forecasts and their application towards clean energy. The secondary objective of this research is exploration of a new evolutionary algorithm as a possible statistical tool to describe some nonlinear aspects of the atmosphere. The products of this work are summarized in four chapters.

Motivated by the difficulty in forecasting montane precipitation for hydroelectricity, a novel model output statistical method is introduced to improve numerical daily precipitation forecasts. The proposed method is gene expression programming (GEP). It is used to create a bias-corrected ensemble, called a deterministic ensemble forecast (DEF), which could serve as an alternative to the traditional ensemble average. Comparing the verification scores of GEP DEF vs. an equally-weighted (traditional) ensemble-average DEF, it is found that GEP DEFs were better for about half of the mountain weather stations tested.

The need for an enhanced electric load forecasting model with better connections to weather variables is addressed next. GEP is used to forecast relative load minima during nighttime and mid-day, and relative load maxima in the morning and evening. A different method is introduced to use GEP to forecast electric load for the next hour. These methods are verified against independent data for a year of daily load forecasts, and are compared against the operational load forecasts archived by BC Hydro, British Columbia's largest electric utility company.

Also, GEP is used to parametrize two non-iterative approximations for saturated pseudoadiabats (also known as moist adiabats). One approximation determines which moist adiabat passes through a point of known pressure and temperature, such as through the lifting condensation level on a skew-T or tephigram. The other approximation determines the air temperature at any pressure along a known moist adiabat, such as the final temperature of a rising cloudy air parcel. This work can be used to better predict cloudy convection in the atmosphere, which can cause hazardous wind gusts at wind turbines, and can drop heavy precipitation in hydroelectric watersheds.

Preface

The main body of this dissertation is composed of work from the following journal manuscripts and a conference poster. The material in these articles has been modified to conform to the dissertation formatting requirements, but the content is otherwise unaltered except for some minor edits.

Chapter 2

Bakhshaii, A. and R. Stull, 2009: Deterministic ensemble forecasts using gene−expression programming. *Weather and Forecasting*, **24**, 14311451, doi:10.1175/2009WAF2222192.1.

The co-author contributions were as follows: Roland Stull proposed the idea of using evolutionary computation in NWP and I applied it as an ensemble averaging tool. The final research was designed cooperatively based on ongoing discussion between Stull and I. The quality-controlled data was provided by Douglas McCollor. The rest of the research was done solely by me. Stull contributed greatly to the writing and editing the manuscript.

Chapter 3

Bakhshaii, A. and R. Stull, 2011: Gene-expression programming−an electrical-load forecast alternative to artificial neural networks. Poster presented at the 91st American Meteorological Society Annual Meeting, Seattle,WA.

The co-author contributions were as follows: I identified the possibility of using GEP as a short-term load forecasting tool. The final research was designed based on ongoing discussion between Roland Stull and me. Data was provided by Heiki Walk from BC Hydro. The research and manuscript writing was done jointly by Stull and me. Stull contributed greatly in editing the manuscript. The model was used to provide operational short term load forecasts during the 2010 Winter Olympics, and I was the person who implemented it.

Chapter 4

Bakhshaii, A. and R. Stull, 2012: Electric load forecasting for W. Canada: A comparison of two nonlinear methods. *Atmosphere-Ocean*, accepted 30 Jan 2012.

The co-author contributions were as follows: I formulated the concept and designed the model. Data was provided by Heiki Walk from BC Hydro. Roland Stull played a large role in rewriting and editing the manuscript.

Chapter 5

Bakhshaii, A. and R. Stull, 2012: Saturated Pseudoadiabats −A Non-iterative Approximation. Submitted for professional peer review on 7 Feb 2012.

The co-author contributions were as follows: Roland Stull identified the need for this research and proposed using GEP. The final research was designed jointly based on ongoing discussion between Stull and me. The research including the data analysis, modelling, and verification has been done solely by me. Stull made large contributions writing and editing the manuscript.

Table of Contents

List of Tables

List of Figures

Acknowledgments

Many thanks to Roland Stull for providing exceptional support. Thanks to Doug McCollor and Heiki Walk of BC Hydro for providing the quality-controlled data set. Thanks to Thomas Nipen, George Hicks, and Henryk Modzelewski for programming aid; to anonymous referees who suggested important improvements; and to colleagues: Bruce Thompson, May Wong, Rosie Howard and Greg West.

Funding was provided by BC Hydro, the Canadian Foundation for Climate and Atmospheric Science, the Canadian Foundation for Innovation, the British Columbia Knowledge Development Fund, The Canadian Natural Sciences and Engineering Research Council (via Discovery, Equipment and Strategic grants) and UBC. The Geophysical Disaster Computational Fluid Dynamics Centre provided computing resources.

Dedicated to my daughter Kiana.

Chapter 1

Introduction

Promise yourself to live your life as a revolution and not just a process of evolution.
— Anthony J. D'Angelo

1.1 From Darwinian evolution to evolutionary computation

Considering evolution without Charles Darwin is impossible. After 150 years of discoveries, he still is one of the most influential figures in science and human history. The influence includes the invention of Darwinism despite the fact that Charles Darwin did not invent a belief system himself. He came with an idea that flourished by later discovery of genes and the mechanism of inheritance. Darwin (1859) coined the term "natural selection" to describe how all species of life have descended over time from common ancestry. Darwin's (1859) Theory of Evolution that is driven by natural selection (brutal selection may be a more proper term nowadays) is a widely held notion. Both natural selection and evolution have made a huge influence on bio-inspired science.

Computational science's fascination with evolution dates back to the invention of computers. Turing's provocative idea of "building a brain" and his 1946 (Turing, 1946) proposal on "genetical or evolutionary search" (Eiben and Smith, 2003) was the beginning of evolutionary computation. He suggested a range of ideas for systems that could be made to modify their own programs. These ideas include his prototype neural networks and genetic algorithms. Bremermann (1962) was the first one who applied his idea as optimization through evolution and recombination (Eiben and Smith, 2003; Bremermann, 1962). Soon after, Turing's ideas were developed into different computational structures in different places: evolutionary programming and genetic algorithms in the USA (Fogel et al., 1966; Holland, 1975), and evolution strategies in Germany (Rechenberg, 1965; Schwefel, 1965). These three branches continued their existence under the evolutionary computation umbrella while new ones were joining the party.

Evolutionary algorithms (EAs) include all methods of simulating evolution on a computer. Regardless of the application, these algorithms are founded on Darwin's evolution principles:

- Finite resources can support only a limited number of individuals.

- Lifeforms have descended over time from common ancestors.
- Competition for resources have very effectively increased the chance of survival of better adapted individuals (selection).
- Variations occur through random changes yielding a constant source of diversity.

Different EAs are different in performance and structure. The next section explains EAs in more detail.

1.2 Evolutionary algorithms

There are many different variants of EAs. The common ideas behind all methods are evolutionary dynamics of reproduction, mutation, competition and selection by generating a population of computer programs and testing them within an environment for which environmental pressure increases adaption by selecting the fittest individual. EAs are conventionally divided into different categories based on how they are represented in a problem:

- Genetic Algorithms (GA) with binary-string representation
- Evolution Strategies (ES) with real-valued vector representation
- Evolutionary Programming (EP) with finite-state machine representation
- Genetic Programming (GP) with expression-tree representation

The classification has mainly historical origins and some new variants of EAs do not follow this division. EAs such as Immune Programming (Musilek et al., 2006) bridges between GP and immune systems (IS). Gene Expression Programming (GEP) uses both GP and GA representations (Ferreira, 2006). Different categories of EA work better for different problems. In this dissertation I use GEP as a function-finding tool. The detailed history and description of GEP is provided in each chapter, according to their different applications.

1.3 From rain drops to light bulbs

Hydroelectric power is a reliable, renewable, clean and inexpensive domestic source of electricity. The relatively low operation and maintenance costs and reliable technology make hydroelectric power very desirable.

Currently, hydropower is the most important and widely-used renewable source of energy in the world. It represents 19% of total electricity production. After China, Canada is the second

largest producer of hydroelectricity, followed by Brazil and the United States (USGS, 2011). British Columbia produces almost 90% of it's electricity by hydropower (BC Hydro, 2012).

Although hydropower is a good source of energy, it is not exempt from resource limitations. Energy management sectors must efficiently balance demands and production and plan ahead to foresee all possible pitfalls. Weather forecasts greatly affect hydroelectric power planning, reservoir operation, energy trading, and social and environmental planning. Many short-term to long-term decisions are extremely dependent on weather forecasts.

Motivated by the need for better hydrometeorological forecasts for the energy sector, this dissertation uses GEP to address three issues related to renewable energy. Chapter 2 presents a novel model-output-statistics (MOS) approach for improving numerical forecasts of montane precipitation driving water inflow into hydroelectric reservoirs.

Electric-load forecasting motivates the next portion of research. The amount of electricity generated and sent through the transmission and distribution lines by utility companies must match the actual electricity consumption (i.e., load) in order to prevent brownouts and power surges. Thus load forecasts are crucial for the operational scheduling of hydroelectric facilities and wind turbines. Based on interviews with BC Hydro load experts, I learned of the large difference between their perceptions of how weather affects load versus how they actually use weather to model load.

Regarding their perceptions, BC Hydro experts state that load depends on temperature, winds, cloudiness, precipitation, humidity, and other weather variables. They also notice that load varies with day of the week, month, holidays, and sometimes with psychological factors (such as when snow is forecast, even if it does not occur). But the computer model for electric load that BC Hydro uses operationally every day is based only on the temperature at Vancouver airport. In effect, the BC Hydro experts start with these single-point temperature-only-based load forecasts, and then manually tweak the load forecasts based on their experience and using other available information.

So motivated, Chapters 3 and 4 present experiments in different ways to objectively forecast electric load. Chapter 3 utilizes many calendar and weather variables at several population centers in BC as predictors in GEP to forecast total electric load served by BC Hydro within British Columbia. This work separates the deterministic (recurring, seasonal) periodic signals from the total load signal over many years, and then trains GEP on the random-looking residual load. The resulting GEP model can then be used to forecast future residual load, which can be added to the future deterministic signal to create load forecasts for every hour in the next day. This approach is verified with a year of independent weather and load data.

Chapter 4 takes the other approach of testing how much of the total electric load can be forecast using only Vancouver temperature as input. A predictor-corrector approach is taken in this research, where GEP and artificial neural networks (ANNs) are alternative nonlinear methods used to predict

the daily variation load as a function of Vancouver temperature. Recurring hourly errors from one day to the next are then corrected with simple, linear bias correction. The result is compared with the operational load forecasts produced by BC Hydro.

The next chapter (Chapter 5) presents an algorithm that can accelerate the computation of saturated adiabatic processes that could be used to improve the forecasts of deep convection such as thunderstorms, which can threaten wind turbines with strong wind gusts, and which can cause intense localized rain events in hydroelectric watersheds. The final chapter summarizes the achievements of the previous chapters.

Chapter 2

Deterministic Ensemble Forecasts using Gene-expression Programming

A method called gene-expression programming (GEP), which uses symbolic regression to form a nonlinear combination of ensemble NWP forecasts, is introduced. From a population of competing and evolving algorithms (each of which can create a different combination of NWP ensemble members), GEP uses computational natural selection to find the algorithm that maximizes a weather verification fitness function. The resulting best algorithm yields a deterministic ensemble forecast (DEF) that could serve as an alternative to the traditional ensemble average.

Motivated by the difficulty in forecasting montane precipitation, the ability of GEP to produce bias-corrected short-range 24-h-accumulated precipitation DEFs is tested at 24 weather stations in mountainous southwestern Canada. As input to GEP are 11 limited-area ensemble members from three different NWP models at four horizontal grid spacings. The data consist of 198 quality controlled observation-forecast date pairs during the two fall-spring rainy seasons of October 2003 to March 2005.

Comparing the verification scores of GEP DEF versus an equally weighted ensemble-average DEF, the GEP DEFs were found to be better for about half of the mountain weather stations tested, while ensemble−average DEFs were better for the remaining stations. Regarding the multimodel multigrid-size "ensemble space"spanned by the ensemble members, a sparse sampling of this space with several carefully chosen ensemble members is found to create a DEF that is almost as good as a DEF using the full 11-member ensemble. The best GEP algorithms are nonunique and irreproducible, yet give consistent results that can be used to good advantage at selected weather stations.

2.1 Bias-corrected ensembles

An ensemble of outputs from different numerical weather prediction (NWP) runs can provide information on forecast confidence, probability of different forecast outcomes, range of possible errors, predictability of the atmosphere, and other metrics. Although expensive ensemble runs are justified predominantly by their probabilistic output (McCollor and Stull, 2009), the most widely used met-

ric is an average of the ensemble members. Except for rare events (Hamill et al., 2000; ECMWF, 2007), this ensemble average usually has better skill, consistency, quality, and economic value than other deterministic NWP forecasts of similar grid resolution (Richardson, 2000; Wandishin et al., 2001; Zhu et al., 2002; Kalnay, 2003). If we define a Deterministic Ensemble Forecast (DEF) as any combination (linear or nonlinear) of ensemble members, then the ensemble average is one type of DEF.

Raw NWP outputs can have large biases in mountainous regions. Biases can be reduced by performing statistical postprocessing; namely, regressing previous forecasts against their verifying observations (the training set), and using the resulting regression equation or algorithm to remove biases in future forecasts (Wilks, 2006). Postprocessing is an important component of the daily operational runs at NWP centers. A large variety of NWP postprocessing methods have been tested over decades by many investigators. Recent work (Gel, 2007; Hacker and Rife, 2007; Yussouf and Stensrud, 2006, 2007; Yuan et al., 2007; Cheng and Steenburgh, 2007; Hansen, 2007) includes multiple linear regression/model output statistics (MOS), sequential estimation of systematic error, updatable MOS, classification and regression trees, alternative conditional expectation, nonparametric regression, running-weighted-mean bias corrected ensemble (BCE), Kalman filtering (KF, a linear approach), neural networks (nonlinear), and fuzzy logic.

DEF (e.g., ensemble averaging) and statistical postprocessing are intertwined. DEF reduces random error, while statistical postprocessing reduces systematic errors (biases). Hence, using both together is recommended to provide an optimum forecast.

An explicit way to combine DEF and statistical postprocessing is to first postprocess each ensemble member individually to reduce its bias, and then average all resulting ensemble members. This bias-corrected ensemble (BCE) method separates statistical postprocessing from ensemble averaging. For example, the University of British Columbia (UBC) operational short-range ensemble uses Kalman filtering (Bozic, 1994) to remove the biases from individual temperature forecasts before weighting them equally in a pooled ensemble average (Delle Monache et al., 2006; McCollor and Stull, 2008a). Each station gets its own Kalman-filter correction. Other investigators have used running means instead of Kalman filters to do the bias correction (Stensrud and Skindlov, 1996; Stensrud and Yussouf, 2003, 2005; Yussouf and Stensrud, 2007; Eckel and Mass, 2005; Jones et al., 2007; Woodcock and Engel, 2005; McCollor and Stull, 2008a).

An example of an implicit combination is a weighted ensemble average, where each ensemble member is weighted inversely to its past forecast error. Here, statistical postprocessing is in the form of the computation of the past error from a training set. Those ensemble members with larger biases will have larger error statistics, giving them smaller weights in the ensemble average−an approach that tends to reduce overall bias of the ensemble average. Yussouf and Stensrud (2007) found

explicit approaches to be slightly better than implicit, while we found (unpublished) nearly identical skill for implicit and explicit equally weighted Kalman filtered (pooled) temperature ensembles at the University of British Columbia (UBC).

Bias-corrected ensemble averages have worked well for variables such as temperature, but are more difficult to implement for montane precipitation perhaps because of its time series with many zeros and its hybrid discrete/continuous frequency distribution [Bernoulli-gamma or Poisson-gamma; see review by Cannon (2008)]. So motivated, we present a new way to create bias-corrected DEFs via the gene-expression-programming (GEP) method of evolutionary programming. Section 2.2 describes the GEP concept. Section 2.3 explains our motivation, and covers case-study data and experimental procedure. Section 2.4 shows case-study results, and Section 2.5 presents a sensitivity test to determine the most important ensemble members. A discussion of the uniqueness and utility of the algorithms found using GEP is given in Section 2.6, with conclusions in Section 2.7.

2.2 Evolutionary programming

2.2.1 Method

The ensemble average uses three operators $(+, *, /)$ to combine the individual ensemble members into a DEF. Using 24-h accumulated precipitation (P24) at any one weather station as an example, the algorithm of the simple ensemble average is $\mathrm{DEF}_{\mathrm{P24}} = [c_1 * \mathrm{P24}_1 + c_2 * \mathrm{P24}_2 + ...c_N * \mathrm{P24}_N]/N$, where N is the number of ensemble members, c are the respective weights, and the subscript indicates the ensemble member.

But suppose we allow more operators, more functions (sine, exponential, if, or, greater than, etc.), and nonlinear as well as linear combinations. An arbitrary example is $\mathrm{DEF}_{\mathrm{P24}} = (\mathrm{P24}_1^{c_1}) * \sin(\mathrm{P24}_2) + \ln(N)/\exp(c_N * \mathrm{P24}_N)$. The result is an equation or algorithm to calculate the DEF. Consider this algorithm to be one individual in a population of algorithms. The other individuals are different algorithms using different sets of functions, constants and operators to estimate the DEF, but each individual uses the same set of N-ensemble P24 inputs, and produces a forecast for the same weather station.

If the algorithms are created somewhat randomly, most individuals would give DEFs that do not verify well against observations. Allow those individuals with higher verification scores to spawn slightly modified daughter versions of themselves (with changes to a small portion of their operators, functions, or constants) to create a new population of algorithms. The individuals in this second generation are again culled via computational natural selection for a "training set" of data.

After many generations, the population evolves to favor the stronger individuals, thereby giving

us the better algorithms; hence, the name evolutionary programming. The process of finding an algorithm that best reduces some statistical measure of error is called symbolic regression. This regression is run separately at each weather station, yielding individual algorithms with different constants and different functional forms at the different stations. After a best individual algorithm appears in the training data set, it can be used with the real-time forecasts for that station. As will be shown, the resulting algorithms are often not simple, yet they can give the best bias-corrected DEF at some of the weather stations.

This approach is motivated by the human body. The body is an amazingly complex system of chemical, electrical, hydraulic, pneumatic, and other physical processes that works exceptionally well. This fact has caused us to re-evaluate Occam's razor, one of the philosophical underpinnings of modern science. Instead of believing that the "simplest explanation tends to be the right one", we suggest "a scientific relationship should not be more complex than needed." Regarding the human body, it is amazingly complex, but not more so than needed.

We suspect that the existing postprocessing and DEF algorithms for most surface weather variables (precipitation, humidity, winds, etc.) are much too simple. The linear approaches of regression and Kalman filtering use only a handful of predictors with roughly an equal number of weights. The math functions they use are only $(+, -, *, /)$. Even the neural network assumes a handful of fixed functions $(+, -, *, /,$ exp or tanh) for the transfer functions between neural nodes, and allows only a fixed large number of weights to be fit during training.

If we consider the mathematical space of all possible algorithms, then the current methods of NWP statistical postprocessing and DEF are sampling just a small portion of this space. Evolutionary programming allows us to sample a much broader portion of algorithm space, and use a much richer set of functions (arithmetic, trigonometric, hyperbolic, relational, etc.) in algorithms that are allowed to grow in complexity as needed to give the best solution.

2.2.2 The evolution of evolutionary programming

Evolutionary programming is a type of machine learning−artificial intelligence (Mitchell, 1997) designed to perform symbolic regression. Three variants in evolutionary programming are "genetic algorithms", "genetic programming", and "gene-expression programming" (Ferreira, 2006). Genetic algorithms (GAs) were devised in the 1960s by Holland (1975), and are coded as symbolic strings of fixed length that work analogous to RNA replicators in nature. Many of the individuals in a GA population evolve into symbolic strings that produce non-functional algorithms, reducing evolutionary efficiency and requiring extra computation to find and remove the nonviable individuals. "Artificial life" (A-life) simulations of Adami (1998) and Lenski et al. (2003) are examples of genetic algorithms, but with a growing computational genome.

Genetic programming (GP) was devised by Cramer (1985) and promoted by Koza (1992, 1994), and Koza et al. (1999, 2003). This method uses nonlinear parse trees of different sizes and shapes, which are analogous to protein replicators in nature. This approach also has the difficulty that many mutations result in non-functional algorithms−wasting computations.

Gene-expression programming (GEP) is promoted by Ferreira (2001, 2006) as the best evolutionary programming method to date. It mimics a fully separated genotype (the genetic information) and phenotype (the physical body as informed by the genetic information) system in nature, but has the advantage that it always forms functional individuals (i.e., always produces valid expression trees in computer code). It also allows unconstrained modification of the genome (allowing wider sampling of algorithm space), allows point mutations, and results in extremely rapid computational evolution.

2.2.3 Gene-expression programming (GEP)

GEP uses a computational data structure analogous to a biological chromosome to hold genetic information for an individual algorithm (Ferreira, 2006). The GEP chromosome is a string of characters, where some characters represent mathematical operators and others represent operands (called "terminals" in GEP). The terminals can be the raw output from NWP models (i.e., the predictors) or numbers (analogous to coefficients or weights). Also, the operators can operate on other operators. This chromosome defines a computational expression tree; namely, an individual algorithm (see Appendix A). Each character in the chromosome defines a node in the expression tree. A chromosome can be designed to represent a single gene (monogenic), or many genes (multigenic). Each gene gives one expression tree. The multigenic expression trees can be combined via a simple function to make the complete individual. Figure 2.1 illustrates an algorithm in (a) normal mathematical form, (b) its computational expression tree, and (c) its representation as a monogenic chromosome in GEP.

Different individuals in the initial randomly created population have different chromosomes, defining different expression trees. Each expression tree is evaluated using all the forecast-observation pairs of data in the training set, and a "fitness" of that expression tree is found using a measure of the forecast error.

Fitness can be calculated using any appropriate statistical measure. We utilize mean absolute error (MAE), root relative squared error (RRSE), and relative absolute error (RAE) (see Appendix B). We modify the fitness measure to nudge solutions toward parsimony by giving slightly higher fitness to those individuals that are simpler. Individuals that are more fit according to one measure are sometimes less fit according to other measures. We chose an optimum fitness measure by trial and error. Also, as the evolution progresses, we get a more accurate DEF by changing the fitness

measure from a statistic that favors outliers (RRSE) to one that does not (MAE).

Individuals that are more fit are given a higher probability of reproducing. This selection is via a computational analogy to roulette-wheel sampling (Goldberg, 1989), where each individual receives a sector of the roulette wheel proportional to its fitness. To maintain population size, the roulette wheel is computationally spun as many times as there are individuals in the population. Although this method gives a greater chance of selection to the fitter individuals, it also allows occasional selection of less-fit individuals, which is important for maintaining genetic diversity in the population.

The combination of input data (forecast-observation pairs), fitness measure, and selection mechanism is called the selection environment. The selected individuals are then reproduced with chromosome modification to form the next generation. Also, the single best individual in each generation is copied without modification into the next generation.

The modification methods of the GEP chromosome include mutation, inversion, transposition, and recombination (see Appendix C). The last method is sexual, because it involves the sharing of genetic information between different individuals within the same generation. Computational evolution is accelerated (relative to biological evolution) by applying these modifications at rates ranging from 0.04 for mutation (e.g., 4 mutations per 100-character chromosome per generation) to 0.4 for one-point recombination. These computational modifications are patterned after observed biological modifications that are known to have been important in driving the evolution of life (Ferreira, 2006).

The process of reproduction, fitness determination, and natural selection is repeated from generation to generation, creating a genetically diverse evolving population. Figure 2.2 illustrates GEP evolution, showing selected generations that provide successively better fits to a simple noisy data set.

The population exists in a computational "world" or environment where the individuals can sexually share genetic information during reproduction. But depending on the complexity of the problem, it is possible that none of the individuals in a world will evolve to an acceptable solution. However, multiple worlds (i.e., separate GEP evolutions) can be created on the computer, each with different random initial populations, and each of which could evolve to different end states. New worlds can continue to be created until one yields an acceptable individual (i.e., an algorithm that works well, within some error tolerance).

An important outcome of this evolutionary approach is that different worlds might yield different acceptable individuals. Namely, there can be an arbitrary number of non-unique good solutions, each of which works nearly as well as the others. Figure 2.3 illustrates this effect, showing non-unique solutions that fit sigmoid data. In statistical postprocessing we are not trying to discover

a physical "law", but instead seek any statistical relationship that works. Uniqueness issues are discussed in Section 2.6.2.

Details of GEP are described by Ferreira (2006). She uses a more sophisticated approach than illustrated above, with a multigenic chromosome that finds successful solutions more quickly. Ferreira implemented GEP in a software package (Ferreira, 2008) that we utilize here.

2.3 Case-study design

2.3.1 Motivation

Montane quantitative precipitation forecasts (QPF) are difficult, yet are critically important for predicting flash floods, avalanches, landslides, and hydroelectric generation. For our short-range limited-area ensemble forecasts we tried Kalman-filter postprocessing and neural-network postprocessing on the individual ensemble members, but both were unsuccessful. The Kalman filter often learns of a QPF bias on a rainy day, and inappropriately applies that bias correction on the next day even if no rain is forecast. The neural network creates a time series with the correct climatological frequency of rain events, but the individual rain forecasts don't match the actual rain events. Also, the neural network doesn't always find the global minimum of its cost function. Moving-averages were found to offer some bias-correction skill for precipitation degree-of-mass-balance in the mountains of western Canada (McCollor and Stull, 2008a). To explore alternatives, we turned to GEP, initially focused on only statistical postprocessing.

While experimenting with GEP, we realized that the optimum algorithm found by GEP was combining the raw NWP precipitation forecasts while simultaneously removing bias. Namely, it was producing a bias-corrected DEF in one step.

The resulting algorithms that GEP created were nothing like we had ever seen−having a complexity that we would never have created by hand. Yet they worked surprisingly well for some of the weather stations we tested. Also, they avoided the over-fitting problem that plagues some neural-network solutions. Based on these enticing results, we designed a more extensive test of this bias-corrected DEF method for many observation stations over many months using multi-model NWP ensemble output, as described below.

2.3.2 Numerical models

The NWP models used at UBC for daily limited-area, short-range (60 h) runs are:

- WRF - theWeather Research and Forecasting model (Skamarock et al., 2005)

- MM5 - the fifth-generation Pennsylvania State University−National Center for Atmospheric Research Mesoscale Model (Grell et al., 1994), and
- MC2 - the Mesoscale Community Compressible model (Benoit et al., 1997)

These models are run in self-nested mode at the following horizontal grid spacings with the following grid designations: Grid 1 (G1) is 108 km, Grid 2 (G2) is 36 km, Grid 3 (G3) is 12 km, Grid 4 (G4) is 4 km (but no Grid 4 for WRF). Thus, on days when all models ran successfully, our multi-model multi-resolution ensemble had 11 members. WRF and MM5 run with two-way nesting, and MC2 with one-way nesting. For the MC2 model, each subsequent finer grid is started 3 hours after the coarser grid, to give time for the numerical solution of the coarser grid to stabilize before starting the finer grid, and to allow the modeled precipitation to spin up.

Initial and boundary conditions for the 108 km runs came from the 00 UTC NCEP-NAM (National Centers for Environmental Prediction - North American Mesoscale) runs. Sixty-hour forecasts are made, but only the 24-hour period between model forecast hours 12 to 36 are used, corresponding to 12 UTC on Day 1 to 12 UTC on Day 2.

2.3.3 Observation data

The case-study area is southwestern British Columbia, Canada, on the Pacific coast of North America (Figure 2.4). This area of complex terrain is characterized by high mountains (many peaks 2,000 to 3,000 m), deep narrow valleys, coastlines, fjords, glaciers, and one large river delta near sea level where metro Vancouver is located. One mountain range forms a spine of Vancouver Island. The higher Coast Range mountains are parallel to Vancouver Island, just east of the Georgia Strait that separates Vancouver Island from the mainland. This region experiences frequent land-falling Pacific cyclones in winter, some with strong pre-frontal low-altitude jets that transport copious amounts of moisture from the tropics (called the "atmospheric river" or "pineapple express").

Twenty-four surface weather stations in this region are used (Figure 2.4 and Table 2.1). Twenty-one of the stations are operated by the BC Hydro hydroelectric corporation as private stations with no ICAO identifier. Three of the stations (CYQQ, CYVR, and CYXX) are operated by Environment Canada, and are abbreviated here as YQQ, YVR, and YXX, respectively. The data set spans October 2003 through March 2005, which includes two Fall-Winter-Spring rainy seasons. Summer data were excluded from this study. Any rainy-season dates with missing observations or missing forecasts were removed, and the remaining forecast-observation pairs were quality-controlled to remove unphysical values. For this case study, if any one or more ensemble members were missing on a date, we removed that date from the data set.

The remaining data set contained 198 days, of which 71% had non-zero precipitation, and (33%,

21%, 9%) had greater than (5, 10, 25) mm/day. The 198-day data set was divided into three unequal sequential portions for training, testing and scoring (Table 2.2). The reason for not interlacing these three subsets is that we wanted to test the best algorithm in simulated forecast mode; namely, for future days (i.e., the scoring set) that might have synoptic regimes different from those in the training and testing periods.

2.3.4 Procedure

The predictand is the difference bias (B24) of 24-h accumulated precipitation (P24) for the MC2 grid 4 forecast at any one weather station:

$$\mathrm{B24} = \mathrm{P24(Observation)} - \mathrm{P24[MC2(Grid\ 4)]} \tag{2.1}$$

The predictors are forecast 24 h accumulated precipitation from all the models and grids (Figure 2.5). Separate GEP runs are made for each weather station, yielding completely different DEF algorithms at each. The 24-h accumulated precipitation variable was chosen for two reasons: (1) it has strong economic value to hydroelectric managers; and (2) it has greater predictability than shorter precipitation intervals (Wandishin et al., 2001; Stensrud and Yussouf, 2007; McCollor and Stull, 2008a,b, 2009).

For any one station, the GEP population is trained using all available data from the first rainy season (107 forecast-observation pairs). These data are called the "training set". After a stable population is reached (namely, after fitness scores plateau during the evolution), all of the surviving individuals are tested against the first half of the remaining data (a "testing set" of 45 to 51 forecast-observation pairs, depending on the station). With such a "testing set" we weed out those individuals that overfit the data or that are unnecessarily complex, allowing a "best" DEF algorithm to be selected. This one best algorithm is then further evaluated (scored) against the remaining independent data (40 forecast-observation pairs, called the "scoring set", see Table 2.2). The resulting verification statistics for the scoring set given here are for the 24-h precipitation, not for the bias B24.

2.3.5 Gene-expression-programming specifications

The GEP software package (Ferreira, 2008) is flexible regarding the size and nature of the chromosomes, the functions available, the creation of numerical constants, the choice of fitness statistic, the mutation and modification rates, etc. Although Ferreira gives some guidelines, the user must experiment via trial and error to find parameters that yield viable populations.

After much experimentation for YVR, we settled on the GEP parameters listed in the "final

setting" column of Table 2.3. The final setting is not necessarily the absolute best one (which might take an infinite number of worlds/runs to discover), but it is one that gives an ensemble average that works well for many of the stations. A set of 80 mathematic and logic functions (see Appendix D) were used to build and modify the chromosomes.

Different choices in allowed parameters allow different DEF algorithms to be created. Even with the same set of GEP parameters, different populations evolve when the GEP run is restarted, because of the quasi-random nature of the initialization, mutation and selection mechanisms. This non-unique and irreproducible nature of GEP is disconcerting at first, but is a characteristic that is irrelevant to the ability of GEP to find useful economically valuable DEFs that work easily and effectively in an operational NWP environment (see Section 2.6).

To illustrate the effect of different GEP parameter settings (Table 2.3) on the DEF algorithm for 24-h precipitation, separate evolutions (worlds or runs) are created for the weather station at Vancouver International Airport (YVR), but with a smaller set of functions ($+, -, *, /$, sqrt, exp, ln, $x^2, x^3, x^{1/3}$, sin, cos, atan) for only the comparison runs prescribed by Table 2.3. After many runs for each setting, the best bias-corrected-DEF algorithms are selected using the training and testing subsets of data, and are then verified against the scoring subset of data. Figure 2.6 compares verification scores for the best algorithms from these different GEP settings. Shown for comparison is the verification for our current operational method, where a "pooled ensemble" (PE) average is created by weighting all ensemble members equally.

Our "final setting" yields a DEF with the best verification scores for all statistics except mean error. It also has better correlation and "degree of mass balance" (DMB). DMB is defined as the ratio of predicted to observed net water mass during the study period (Grubisic et al., 2005) − an important measure for hydrometeorologic studies (see Appendix B). Changing the fitness function from the root relative square error (RRSE) to the relative absolute error (RAE) verifies significantly worse (compare settings I and II in Figure 2.6). Also, the size of genome (the allowed complexity of the algorithm) is very important. The effect of genome size can be examined by changing the number of genes or the length of genes (compare settings I and III).

Overall, the final setting yields DEFs that verify better than other settings we tried (not shown here). This does not mean it is the absolute best setting; however, it is better than the others we tested. These "final settings" are used for all case-study and sensitivity experiments, described next.

2.4 Results

2.4.1 Verification for all 24 weather stations

In this case study, we performed GEP evolutions separately for each of the 24 weather stations, yielding different "best" DEF algorithms for each station (see Appendix E). As described in the previous section, training and testing portions are used to find the best DEF algorithm, and the scoring portion is used for all verification statistics below.

Detailed verification statistics are shown in Figure 2.7, and a concise tally of results is in Figure 2.8. The different statistical measures give different indications as to which DEF approach (GEP or PE) is best. Since RMSE (root mean square error) is very similar to the RRSE (Root Relative Square Error) statistic that was used in the "training" set to determine fitness, it is anticipated that RMSE will show good GEP results at the largest number of stations. The RRSE fitness measure favors extreme precipitation events.

GEP gives lower (better) RMSE than the classical PE method for 11 of the 24 stations and is nearly tied for 10 stations, while the PE method is better for 3 stations. Using the mean absolute error (MAE) metric, only 6 of the 24 stations have lower error using GEP, 5 stations are nearly tied, and 13 stations have lower error using PE. Although some stations were not well fit by the GEP algorithms that were found using the "final" settings from the previous section, it is possible that better GEP DEF algorithms could be found with different settings and variables. There is no reason that a single set of GEP settings should work the best for all stations, so future work will examine optimizing the GEP settings separately for each station.

These results suggest that different DEF methods are best at different stations, with no clear winner for this case-study data set. Using an overall tally as a crude measure, Figure 2.8 has 41 black tiles (GEP best), 41 white tiles (pooled-ensemble best), and 38 grey tiles (nearly equal verification). Comparing the stations in Figure 2.8 with their locations in Figure 2.4, one finds that the stations for which GEP DEFs are best are scattered over a wide range of terrains and distances from the coast. There is a cluster of good GEP stations in the Lower Fraser Valley (YVR, YXX, and ALU) the metro Vancouver region.

2.4.2 A closer examination of Vancouver Airport station (YVR)

Vancouver Airport station gets special scrutiny because of the large number of people (2 million people) in metro Vancouver who can be impacted by urban flooding and landslides triggered by heavy-precipitation events. Figure 2.9 shows 24-h precipitation amount observed at Vancouver International Airport (YVR) over a period spanning the two rainy seasons. Figure 2.10 (data points)

shows the precipitation bias (the predictand) of the MC2 model at 4 km grid spacing for that same period. Also shown in Figure 2.10 is the GEP estimate of 24-h precipitation bias. One component of our implementation is that whenever the GEP bias-corrected forecast gives a negative precipitation amount, we truncate the amount to zero. Thus, those "apparent" erroneously large negative biases (B24) of the GEP forecasts (in Figure 2.10 a and b) do, in fact, give accurate precipitation amount (P24) forecasts after truncation.

A comparison of DEF precipitation amount (P24) forecasts is shown in Figure 2.11a for the scoring data subset at Vancouver Airport (see Appendix F). To get these results, GEP bias forecasts (Figure 2.5a) are applied to the raw MC2(Grid 4) forecasts using Eq. 2.1 to get forecasts of precipitation amounts (Figure 2.5b). Figure 2.11b shows large variability among the raw NWP forecasts (ensemble members, used as predictors) for the heavy rain event in Figure 2.11a. For this rain event on 19 January 2005, the pooled ensemble better fits the observation than the GEP forecast, while during 20 January−5 February 2005 the GEP forecasts fit better (Figure 2.11a and b). For future research, we will use GEP to first classify the predicted precipitation (extreme, normal, none) before finding the best DEF QPFs for each category.

2.5 Sensitivity studies to identify the most important ensemble members

The DEF algorithms devised by GEP to combine all the ensemble members are very complicated (see Appendix E). While they utilize all 11 ensemble members as predictors, it is not obvious from the equations which members have the largest influence on the DEF. One way to address this is via simple sensitivity studies.

Keeping the GEP algorithm fixed for each station, we input a large error for one of the predictors (i.e., one of the ensemble members), and measure the resulting verification statistic for P24. If the ensemble member was relatively unimportant, then a large error in its P24 value will have only a small effect on the overall verification. However, if the ensemble member is an important contributor to the overall GEP forecast, then introduction of a large error in the input ensemble member will result in a substantial increase in verification-statistic error for the whole GEP forecast.

As an example of this sensitivity study, we halve and double the ensemble-member inputs one at a time, and record the resulting verification statistics (Figure 2.12) for Vancouver Airport. (Because the GEP algorithm can be highly nonlinear, doubling and halving could yield different conclusions. As it turns out, halving and doubling gave similar results.)

We find that a small number of "important" ensemble members dominate the DEF outcome. The set of "important" predictors has the following characteristics: (1) many of the models are

included (MC2, MM5, WRF); (2) a range of grid sizes are included (but not necessarily from the same model); and (3) the set contains a small number of elements that satisfy characteristics (1) and (2). For example, Figure 2.12 shows that the following predictors were important for a good GEP DEF algorithm at YVR: WRF(108 km), MM5(36 km), MM5(108km) , and MC2(4 km).

A fourth characteristic is that the set of "important" predictors often changes from station to station, but continues to satisfy characteristics (1) - (3). This difference is not surprising, given that GEP gives different algorithms for each different station. To illustrate, Table 2.4a and b show the "ensemble space" of our case study, and compare the important ensemble members for Vancouver Airport (YVR) and for Comox (YQQ).

Thus, the DEF that is most efficient (providing the maximum information per fewest ensemble members) is gained by spanning all grid sizes (which helps compensate for location errors of synoptic systems, and for terrain-related errors) and spanning all models (maximizing the variety of model physics, dynamics and discretizations). Any additional ensemble members for models or grids already covered by the "important" set offer marginal new information. Namely, not all of the ensemble members in a multi-model multi-grid-size DEF provide independent data. Thus, an efficient DEF spans the ensemble space sparsely. Similar conclusions have been found using best-member diagrams (Hamill and Colucci, 1997; Charles and B.A., 2009) and Bayesian model averaging weights (Raftery et al., 2005).

To examine the robustness of this result, we ran additional GEP evolutions where the ensemble members were reduced from all 11 to the 3 most-important ones indicated in Table 2.4a. The DEF verification scores (Figure 2.13) at YVR using the three "best" ensemble members are only slightly degraded from the scores using all 11 members.

We repeated the experiment with a variety of different 3 "best" ensemble members (based on other individuals from the GEP run that were almost as good as the best), and also based on the pooled ensemble (with 3 and with 11 members, see Figure 2.13). For this case study at YVR, all GEP DEFs (with 3 or 11-member ensembles) give better forecasts than all pooled ensembles (with 3 or 11 members). These results support the hypothesis that sparse spanning of ensemble space can capture most of the signal, if the ensemble members are chosen wisely.

In future work, we will examine the weaker ensemble members to determine what aspects of the NWP dynamics or physics were inadequate, with the aim of suggesting improvements to the NWP codes.

2.6 Discussion

2.6.1 Deterministic ensemble forecasts vs. ensemble averages

When GEP is used to combine the ensemble members into one best deterministic ensemble forecast (DEF), the resulting algorithm is mathematically nothing like an "average" (see Appendix E). Hence, the ensemble average is just one of many ways to combine ensemble members into a deterministic forecast.

Probability and ensemble-spread information is lost when the GEP creates the single deterministic forecast. However, there are other ways to use GEP to preserve ensemble spread information. For example, we could use it for statistical postprocessing each ensemble member separately. Then, the resulting GEP-corrected ensemble members can be combined into probabilistic forecasts using traditional ensemble-averaging methods or using nonlinear combinations.

2.6.2 Reproducibility, uniqueness, and complexity

GEP creates irreproducible algorithms. Namely, different GEP runs for the same station, same training set and using the same GEP parameters will evolve toward different DEF algorithms, the best ones of which verify nearly equally well. This is a byproduct of intentionally introducing random numbers into the specification of the initial population and into the mutation and survival of competing algorithms as they evolve. For modern science founded on experiments that can be reproduced in different labs, the irreproducibility of GEP algorithms can be disconcerting. But this irreproducibility is an intrinsic nature of an evolution (Darwin, 1859).

A reassuring outcome is that the verification scores of the best individual DEFs from different runs are often nearly identical. Namely, each different DEF algorithm is able to extract from the predictor whatever information is available that can be mapped onto the signal of the predictand. Hence, each successfully evolved DEF algorithm has nearly equal utility−they can improve the weather forecasts. In this sense, the successful algorithms are reproducible in their ability to produce the same outputs (within some error tolerance) for identical inputs, even though their internal make-ups are different and irreproducible.

GEP produces non-unique algorithms. This is illustrated in Figure 2.3, where curves (a) and (b) had different sets of functions available during evolution, and evolved toward radically different algorithms. Yet each fit the noisy data equally well. This robustness in the face of non-uniqueness highlights the power of evolutionary computing, and again points to the utility of the end result.

GEP produces DEF algorithms that are much more complex than most humans would have created. Part of this complexity is artificial and redundant, such as illustrated by the GEP equation

in Figure 2.2b. In that example, the equation contains factors of the form $x/(x+x)$, which is how that GEP run created a needed constant value of 0.5. Different runs might have created that constant with different formulas. Although we manually simplified that equation (Figure 2.2b) into a form pleasing to the human eye, the GEP equation is just as valid, and both yield the same outputs for the same inputs (except that more round-off errors can accumulate when unnecessary computations are performed). Even when parsimony pressure is used to reward the simpler algorithms with greater fitness scores, the procedure to reach the final "simplified" algorithm is nonetheless evolutionary, and only sometimes reaches the simplified equation that a human might have created. The cost of this redundant complexity is increased computation time every day when it is used to compute the DEF.

But another part of the complexity is valuable, because it might fit the desired signal better than a simpler algorithm. In spite of this value, the complexity can be scary, as shown in the Appendix E for YVR.

The decision of when to end a GEP run is arbitrary−another disconcerting characteristic. In some simple situations, such as for the noisy but simple data in Figure 2.2 and Figure 2.3, the evolution often reaches the maximum fitness possible without overfitting (i.e., without trying to fit the noise). In this case, it is obvious when to stop the evolution. But for more complex and realistic meteorological problems, the point at which to end the evolution is not obvious. When the evolution starts, significant improvements to the "best" individual algorithm are frequent and obvious. But as evolution proceeds, improvements happen less frequently, and often contribute less increase in fitness. After a 1,000 or 10,000 additional generations with no change, one might be inclined to end the GEP run, even though there is a chance that the 10,001st generation might have offered significant improvement. If the resulting DEF algorithm is good enough (i.e., has small enough error in the independent scoring data set) then we are done. If the error is still large after stopping the evolution, then a recommended approach is to start a new GEP run from scratch; namely, create a new world that will evolve the population of algorithms to a different ending state. Ferreira (2006) thoroughly discusses the probability of reaching a useful ending state via multiple GEP runs.

2.6.3 Operational NWP usage of complex GEP algorithms

Is this postprocessing method worth the added complexity? This is a subjective question for which different users will reach different conclusions. For our own daily operational NWP at UBC, it is worth the added complexity, but only at those weather stations were GEP gives better results. After spending hours of computations daily on hundreds of processors to produce raw NWP output, it is trivial to spend seconds of computation on a couple processors to solve the "best" DEF algorithm (previously evolved by GEP) to improve forecasts at hundreds of weather-station locations in British

Columbia. The amount of forecast improvement directly translates into improved ability to predict potential hydroelectric generation, flash floods, snow avalanches, and water-driven landslides in the mountains.

What about the time to create the evolved algorithm in the first place? The algorithms are created once, but used many times. Even if they need to be re-evolved every season for every city, the utility is large compared to the expense and time needed to create them. Also, the GEP runs can be automated. Again, it is a subjective decision; at UBC, the value is worth the expense.

How will GEP perform for other types of ensembles? We tested only multi-model ensembles here, which might be capturing more of the initial uncertainty in the forecast. For single-model ensembles with multiple initial conditions, the performance of GEP is untested.

2.7 Conclusions and future work

For our case-study data, we find that GEP DEFs are better than equally-weighted ensemble averages of precipitation for about half of the mountain weather stations tested. A sparse sampling of the multi-model multi-grid-size "ensemble space" spanned by the ensemble members can create a DEF almost as good as the full 11-member ensemble. The best DEF algorithms found using GEP are non-unique and irreproducible, yet can give consistent results that can be used to good advantage at operational forecast centers. In spite of the relative complexity of the DEF algorithms created using GEP, they are non-iterative, and thus computationally efficient to use.

For some weather stations, the DEF algorithms found by GEP do not give better results than from the pooled (equally weighted) ensemble average. There is no law of computing that says you must use the same bias-corrected ensemble methods at all weather-station locations, so we recommend the use of different methods at different stations as appropriate.

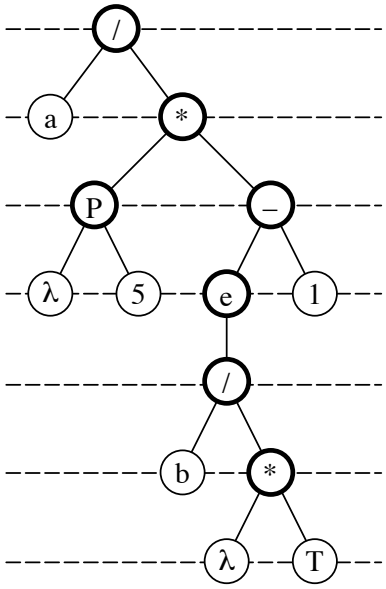
We plan the following future research. We will test GEP DEFs over additional data sets, other locations, and for other weather elements. We will include as predictors the raw NWP precipitation forecasts from neighboring grid points and at neighboring times, to compensate for systematic timing or location biases in cyclone landfall. We will also test using NWP forecasts of other weather elements (temperature, humidity, winds, etc) as predictors in the DEF algorithm for precipitation, and will explore the use of different GEP parameter settings at different stations. We will also build on the work of Cannon (2008) and others to explore whether GEP can be used in two stages: first to classify the forecast precipitation (extreme, normal, or none), and then to use different DEF algorithms for the extreme vs. normal events.

In addition to finding DEFs, we will apply GEP to each ensemble member individually strictly as a bias-correction method, yielding a set of ensemble members that can be used for probabilistic forecasts. We will compare GEP DEFs with additional regression and ensemble-averaging ap-

proaches, such as Bayesian Model Averaging. We will examine the weaker ensemble members to determine what aspects of the NWP dynamics or physics were inadequate, with the aim of suggesting improvements to the NWP codes. All of this work has the ultimate goal of improving the skill of weather forecasts over complex terrain.

(a) $$E = \frac{a}{\lambda^5\left[\exp\{b/(\lambda T)\} - 1\right]}$$

(b)



(c) `/a*P–λ5e1/b*λT...`

Figure 2.1: Different representations of an algorithm, as illustrated using Planck's Law. (a) Mathematical representation, where E is the radiant flux, a and b are constants, T is absolute temperature, and λ is wavelength. (b) Expression-tree representation, where P raises the first argument to the power of the second, e is the exp function, $*$ is multiplication, $/$ is division, and $-$ is subtraction. All circles are nodes, and the thinner circles are terminal nodes. (c) Chromosome representation in GEP, which can be created by reading the expression tree (b) like a book: from left to right on each horizontal (dashed) line, started at the top line and working down. Each function or operator has a known arity (number of arguments), which is used when interpreting the chromosome (the genotype) to build the proper connections between nodes in an expression tree (the phenotype). The "..." represents additional elements of the chromosome that are not read (i.e., do not contribute to the expression tree).

Generation	GEP Chromosome	GEP Equation	Simplified Equation
a) 14	`/ab...`	$y = a / b$	$y = 0.4452$
b) 68	`*x//a-+axxx...`	$y = x \left[\frac{\left\{ \frac{a - x}{x + x} \right\}}{a} \right]$	$y = 0.5 + 0.119\,x$
c) 3967	`*x//a-+axx+/x*-+x//xababx...`	$y = x \left[\frac{\left\{ \frac{a - x}{x + \frac{(x + a)x}{(b/a) - (b/x)}} + x \right\}}{a} \right]$	$y = \frac{(x + 3.7856)/3.7856}{2 + 0.466x\left(\frac{x - 3.7856}{x + 3.7856}\right)}$

d)

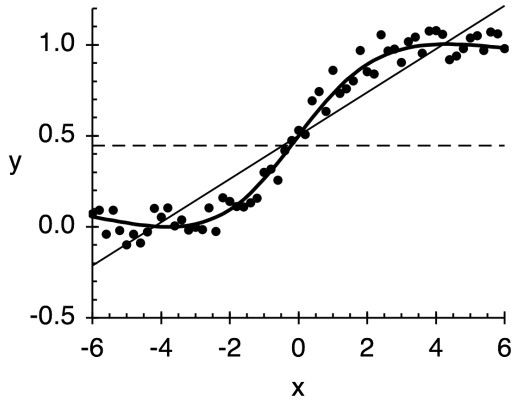


Figure 2.2: Examples of a few of the fittest individuals taken from a world containing 50 monogenic GEP individuals as they evolve toward fitting sigmoid artificial data dots in (d). These artificial data were created using an error function $y = 0.5 * [1 + \text{sign}(x) * \text{erf}(|0.5x|)]$ with added random noise uniformly distributed within $\pm$ 0.1. For this illustration, the set of functions available to GEP is limited to $(+, -, *, /)$, there are only 2 randomly evolving constants (a, b), the GEP chromosomes each have a head of 20 characters and tail of 21, and fitness is determined using root relative squared error (RRSE). Shown for selected generations is the best GEP chromosome within that generation, the equation as literally described by that chromosome, and a manually simplified version of the equation. (a) By generation 14, a reasonable constant y value was born (dashed line), where $(a, b) = (-4.2014, -9.4366)$. (b) By generation 68, a sloping straight line was born (thin solid), where $(a, b) = (-4.2014, -9.4366)$. (c) By generation 3967, a well-fit sigmoid curve was found (thick curved line) with relative explained variance of $r^2 = 0.979$. The two constants had evolved to $(a, b) = (-3.7856, -8.1229)$. The evolution took 5 minutes on a dual-core laptop computer.

ID	NAME	LAT (°)	LONG (°)	ELEVATION (m)	LOCATION
ALU	Alouette Dam Forebay	49.29	122.48	125	Lower Mainland /Interior
ASH	Alsie Lake	49.43	125.14	340	Vancouver Island
BCK	Bear Creek Reservoir	48.5	123.91	419	Vancouver Island
BLN	Brolorne Upper	50.8	122.75	1920	Lower Mainland /Interior
CLO	Clowham Falls	49.71	123.52	10	Lower Mainland /Interior
CMU	Upper Cheakamus	50.12	123.13	880	Lower Mainland /Interior
CMX	Comox Dam Forebay	49.64	125.09	135	Vancouver Island
CRU	Cruickshank River	49.57	125.2	150	Vancouver Island
ELA	Elaho River	50.22	123.58	206	Lower Mainland /Interior
ELK	ELK River above Campbell Lake	49.85	125.8	270	Vancouver Island
GLD	Gold River near Ucona River	49.71	126.11	10	Vancouver Island
GOC	Gold Greek	49.45	122.48	794	Lower Mainland /Interior
HEB	Heber River near Gold River	49.84	125.98	215	Vancouver Island
LJU	Downtown Upper	50.86	123.18	1829	Lower Mainland /Interior
MIS	Mission Ridge	50.75	122.24	1850	Lower Mainland /Interior
NTY	North Tyaughton Creek	51.13	122.76	1969	Lower Mainland /Interior
SCA	Strathcona Dam	49.98	125.58	249	Vancouver Island
STA	Stave River above Lake	49.55	122.32	330	Lower Mainland /Interior
STV	Upper Cheakamus	49.63	122.41	930	Lower Mainland /Interior
WAH	Jones Lake	49.23	121.62	641	Lower Mainland /Interior
WOL	Wolf River Upper	49.68	125.74	1490	Vancouver Island
YQQ*	Comox	49.72	124.9	23	Lower Mainland /Interior
YVR*	Vancouver INTL ARPT	49.18	123.17	3	Lower Mainland /Interior
YXX*	Abbotsford ARPT	49.03	122.37	58	Lower Mainland /Interior

Table 2.1: Twenty-four surface weather stations in Vancouver Island and Lower Mainland with data for the period Oct 2003 through Mar 2005. All stations are operated by BC Hydro except the three Environment Canada stations indicated with "*" in the ID column.

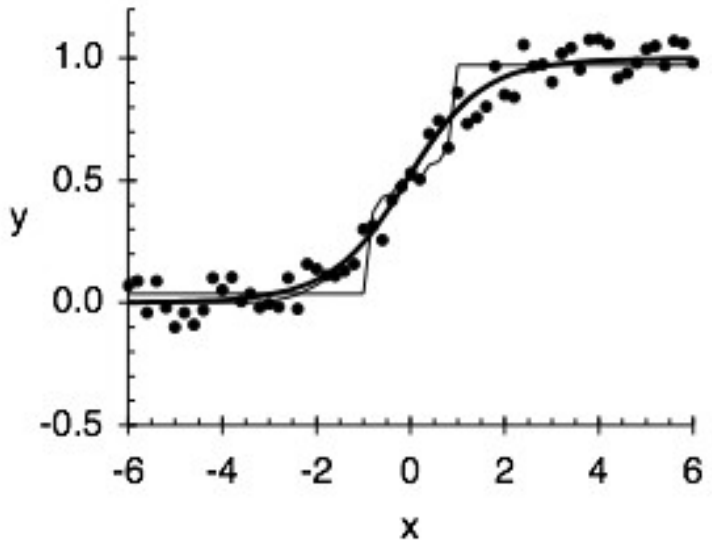


Figure 2.3: Examples of nonunique fits by GEP to the synthetic sigmoid data (dots) from Figure 2.2 . Fitness is measured by the relative explained variance, r^2. (a) When the GEP set of allowable functions is limited to $(+,-,L)$ where L is the logistic function $L(x) = [1+\exp(-x)]^{-1}$, then GEP evolves to a fittest individual (thick line) of the form $y = L[-0.38745 + x + L(x)]$, with fitness $r^2 = 0.978$. (b) If GEP functions are limited to $(+,-,$ Power), then the fittest individual (thin dashed line, mostly hidden behind the thick line) is $y = 0.762055^{2.388183[(0.881012^x)-x]}$, with $r^2 = 0.978$. (c) If the GEP functions are limited to $(+,-,$ mod) where "mod" is the floating-point remainder (as defined in the Fortran 95/2003 standard, not as defined in Microsoft Excel), then the fittest individual (thin solid line) is $y = 50.5057 + \mathrm{mod}\{\mathrm{mod}\{\mathrm{mod}[x,(-0.468079 - x)],(-0.468079 + x)\},x\}$, with $r^2 = 0.952$. These last two examples show how evolutionary computation inexorably approaches a best fit, even when it is restricted to an unusual set of functions that a human would not have intuitively used to fit a sigmoid shape.

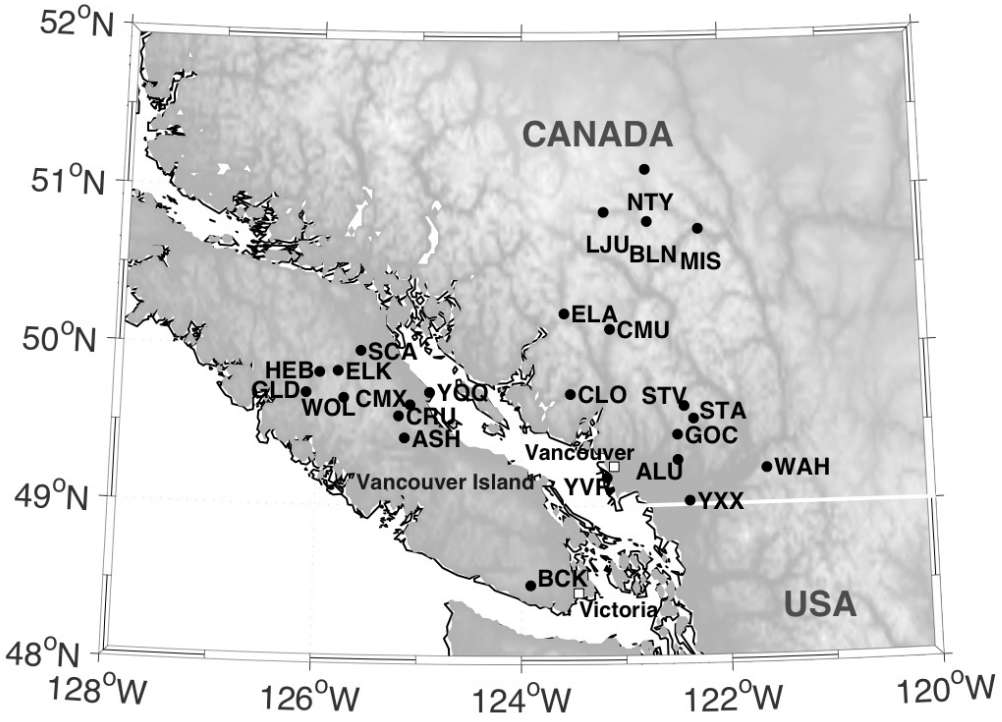


Figure 2.4: Location of weather stations in southwestern British Columbia, Canada.

	NUMBER OF EVENTS	STARTING DATE	ENDING DATE
Training set	107	2003/10/11	2004/10/11
Testing set	47 – 51	2004/10/12	2004/12/31
Scoring set	40	2005/01/01	2005/03/11

Table 2.2: Data subsets. The quality-controlled data set contains 198 days, which is divided into three unequal portions for training, testing and scoring. These three sets help us to test the best algorithm in a simulated forecast mode; namely, for future months that might have synoptic regimes and global climate variations different from those in the training period.

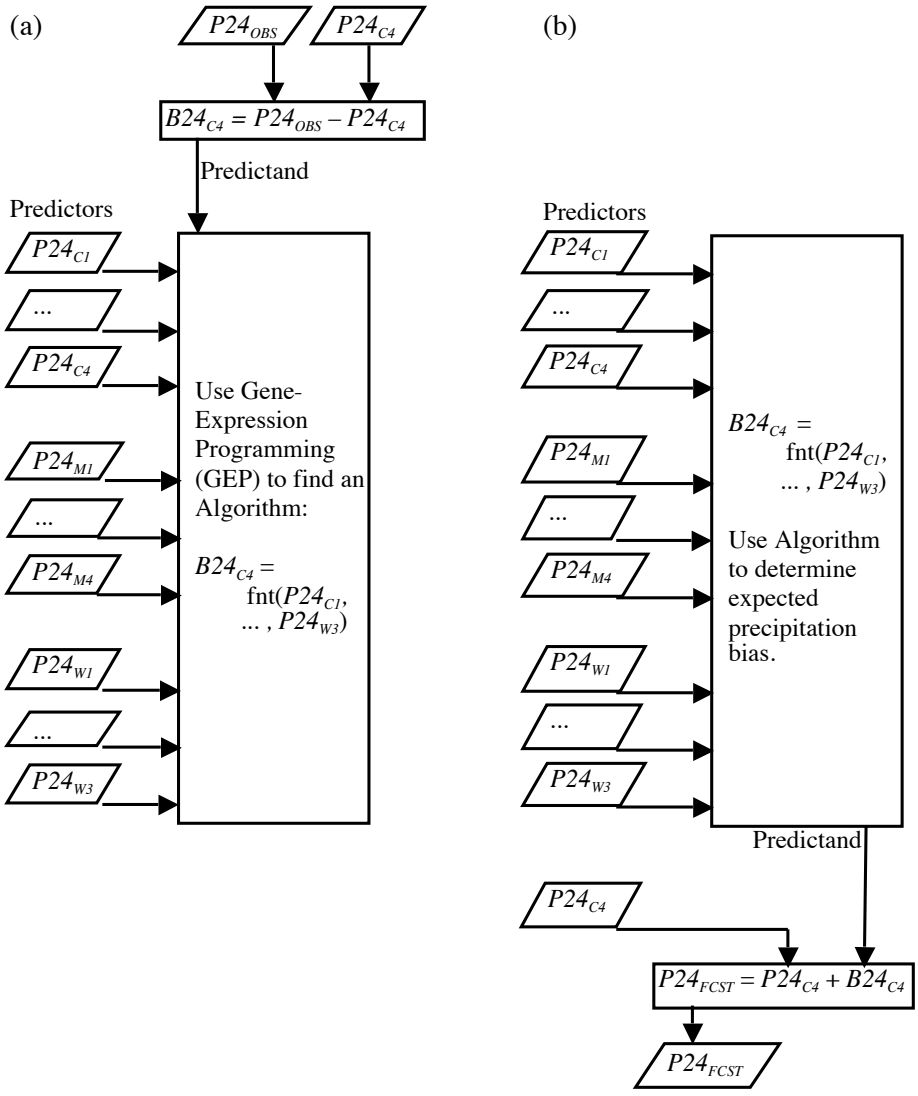


Figure 2.5: Road map of the DEF procedure, done independently at each weather station. (a) First, use GEP on a training set of data to find an algorithm for the bias B24 of 24-h accumulated precipitation. Predictors are raw 24-h precipitation forecasts (P24) from 11 ensemble members, where subscripts (C,M,W) represent the (MC2, MM5, WRF) models, and subscripts $1-4$ indicate Grids $1-4$. Predictand is the bias for only one of the ensemble members: MC2 Grid 4 (corresponding to subscript C4). (b) Then, for the testing and scoring data sets, use the algorithm to create a single deterministic forecast $P24_{FCST}$ from all the ensemble members. $P24_{OBS}$ is the observed 24-h precipitation.

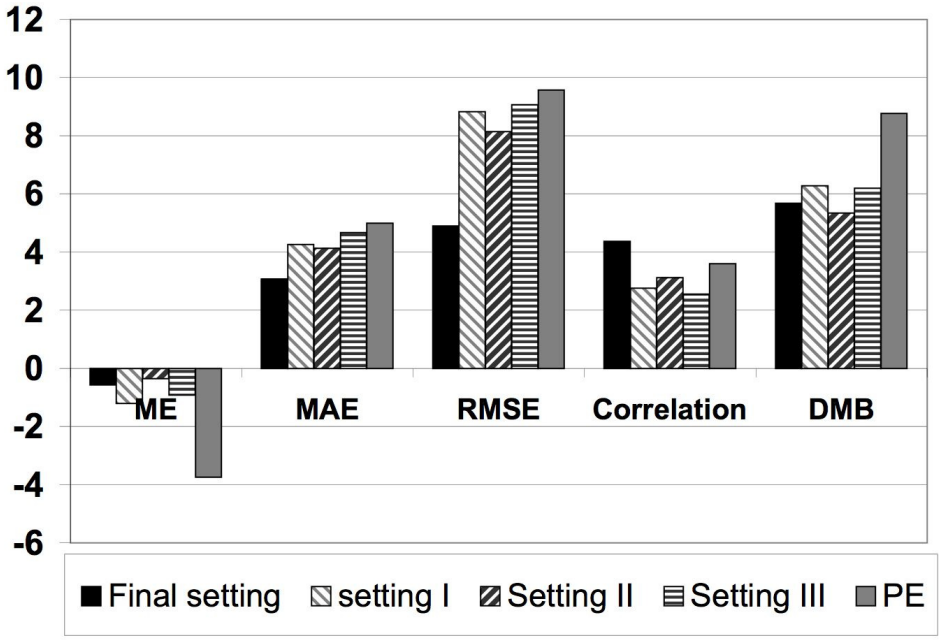


Figure 2.6: Verification statistics for GEP deterministic ensemble forecasts at Vancouver Airport, for the "scoring" subset of data. Four different GEP parameter settings (see Table 2.3) are shown: the final setting (black), setting I (light grey downward diagonal stripes), setting II (dark gray upward diagonal stripes), setting III (dark grey horizontal stripes). Shown for comparison are statistics for the equally-weighted-average pooled ensemble (grey). Plotted are 24-h accumulated precipitation mean error ME (mm), zero is best, mean absolute error MAE (mm), zero is best, and root mean square error RMSE (mm), zero is best. The correlation coefficient between forecasts and observations (r), and degree of mass balance (DMB) have been multiplied by 5 to make them more visible on the plot; hence values closer to 5 are best.

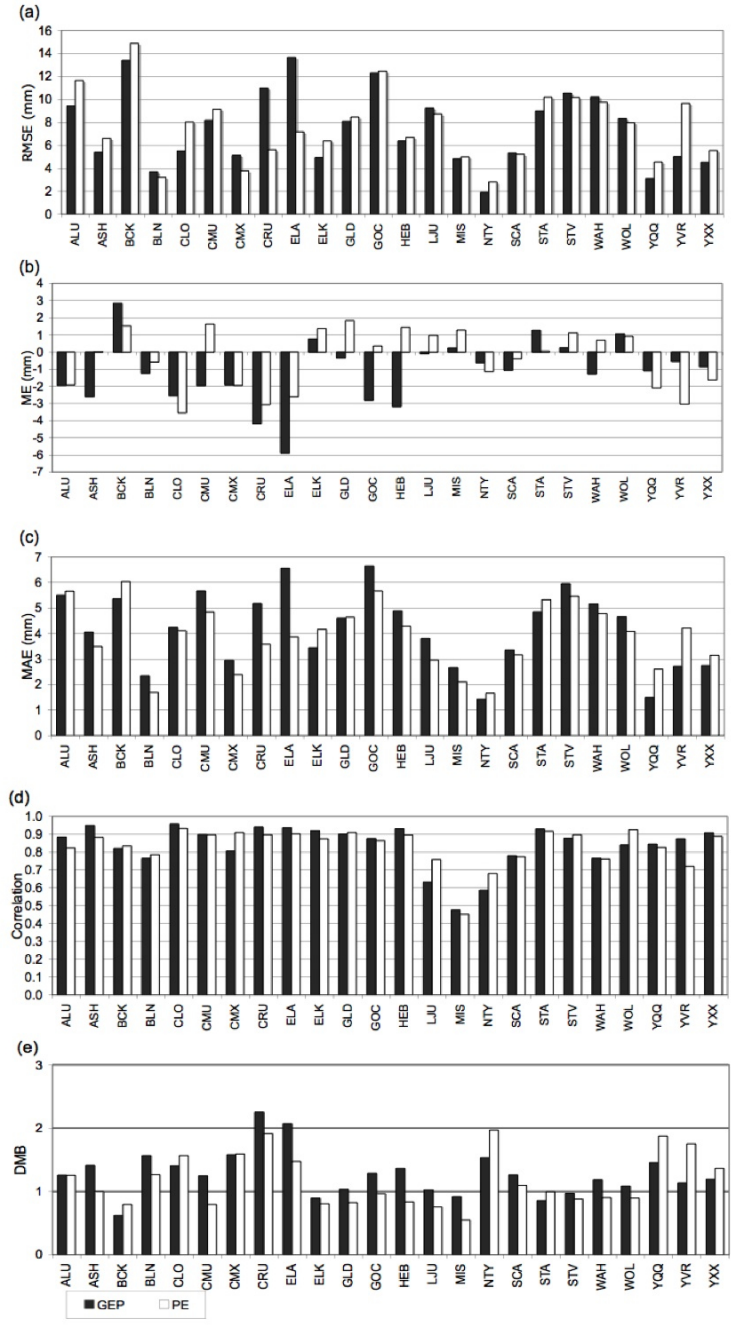


Figure 2.7: Verification statistics of 24-h accumulated precipitation forecasts for the 24 stations in southwestern British Columbia. Black is for GEP, and white for PE (pooled ensemble of equally weighted ensemble members). (a) Root mean square error (RMSE). (b) Mean error (ME). (c) Mean absolute error (MAE). (d) Correlation coefficient (r). (e) Degree of mass balance (DMB). For (a) - (c) an error of zero is best. For (d) and (e) a value of one is best.

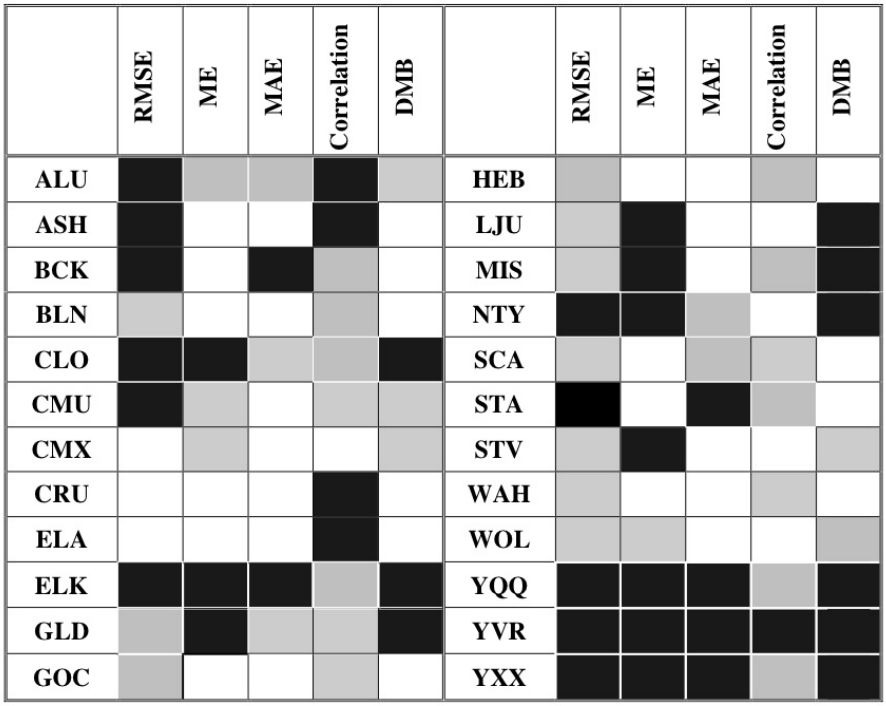


Figure 2.8: Comparison between GEP and PE verification statistics at all 24 weather stations. Black indicates situations where GEP gives a better result; grey indicates no clear winner; and white represents situations where PE gives a better result.

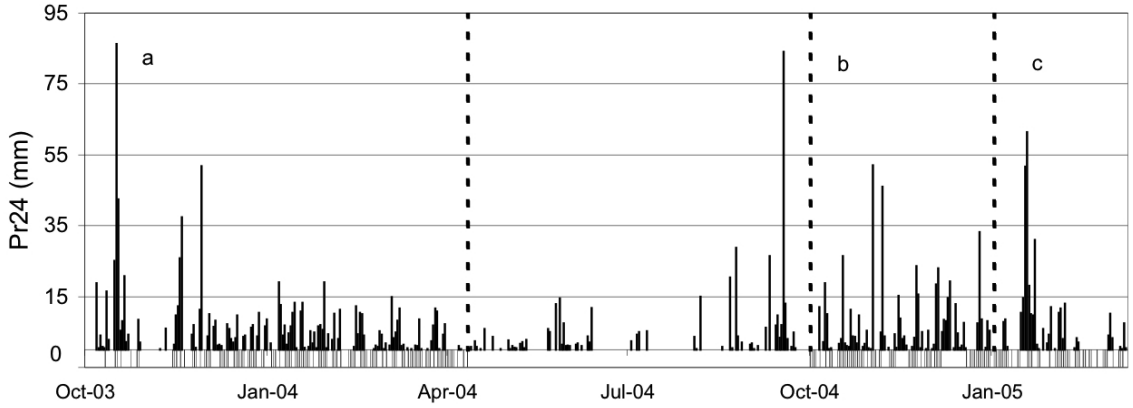


Figure 2.9: Time series of 24 h precipitation amount observations for Vancouver Airport (YVR) during the period 10 Jan 2003 to 15 Mar 2005. Grey lines at bottom show all the dates of available pairs of observations and forecasts. Dashed lines divide the data set to three subsets: (a) training, (b) testing, and (c) scoring. The unlabeled section between (a) and (b) is the drier summer season, not part of this study.

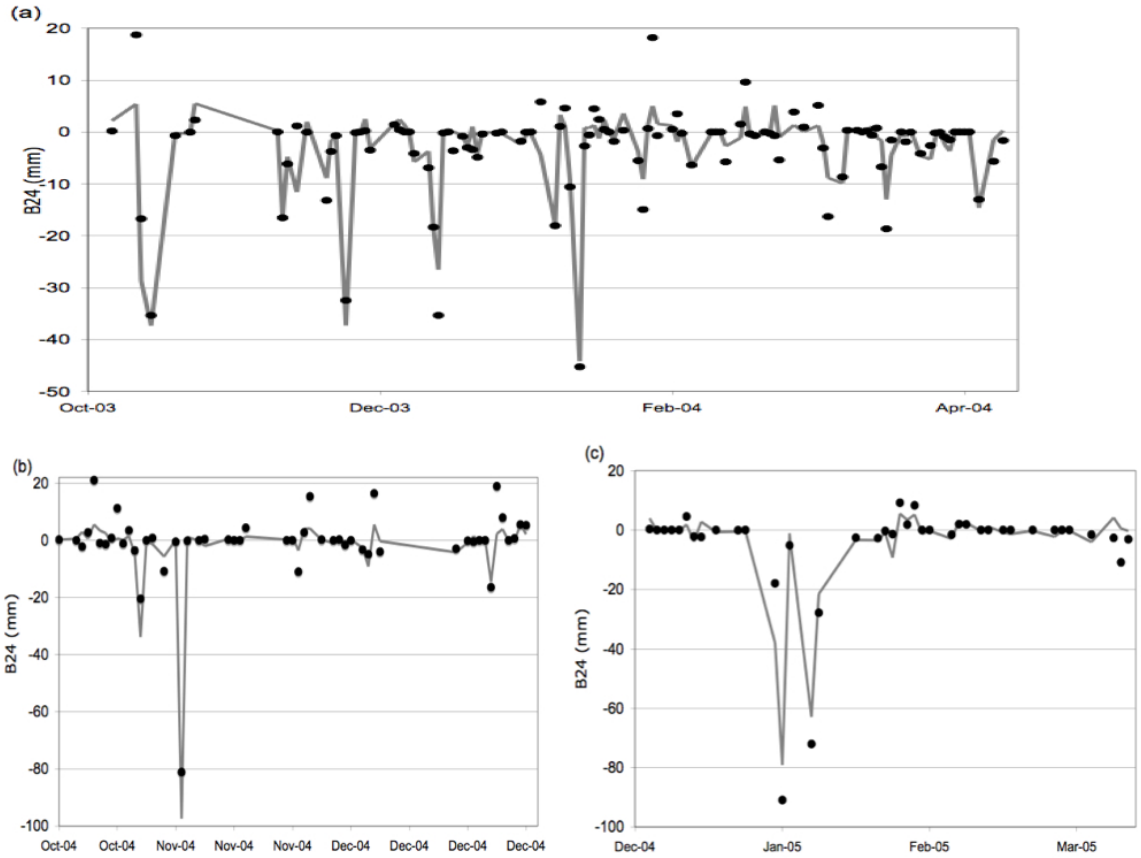


Figure 2.10: GEP ensemble estimates (grey line) of the forecast bias B24 (data points) as defined by equation Eq. 2.1, for the: (a) training, (b) testing, and (c) scoring subsets of data for Vancouver Airport (YVR).

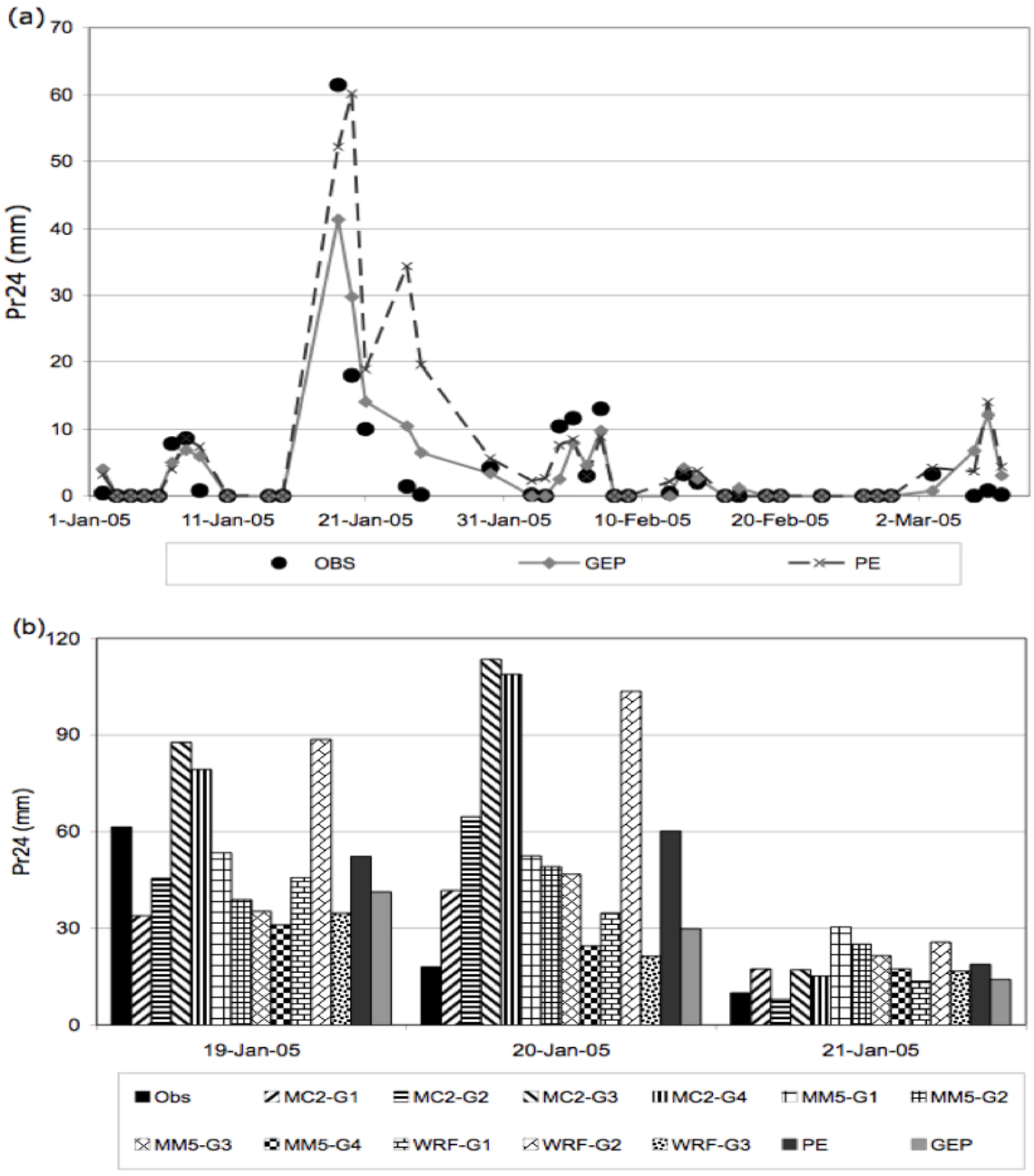


Figure 2.11: (a) Forecasts of 24-h accumulated precipitation amount for YVR from the NWP pooled ensemble (dark grey dashes with crosses) and GEP (light gray line and diamonds). Dots show observed precipitation amount, P24. (b) Illustration of forecasts from each of the ensemble members for three days during the peak rain event. Also plotted in (b) are the observations and two different DEF formulations (PE and GEP).

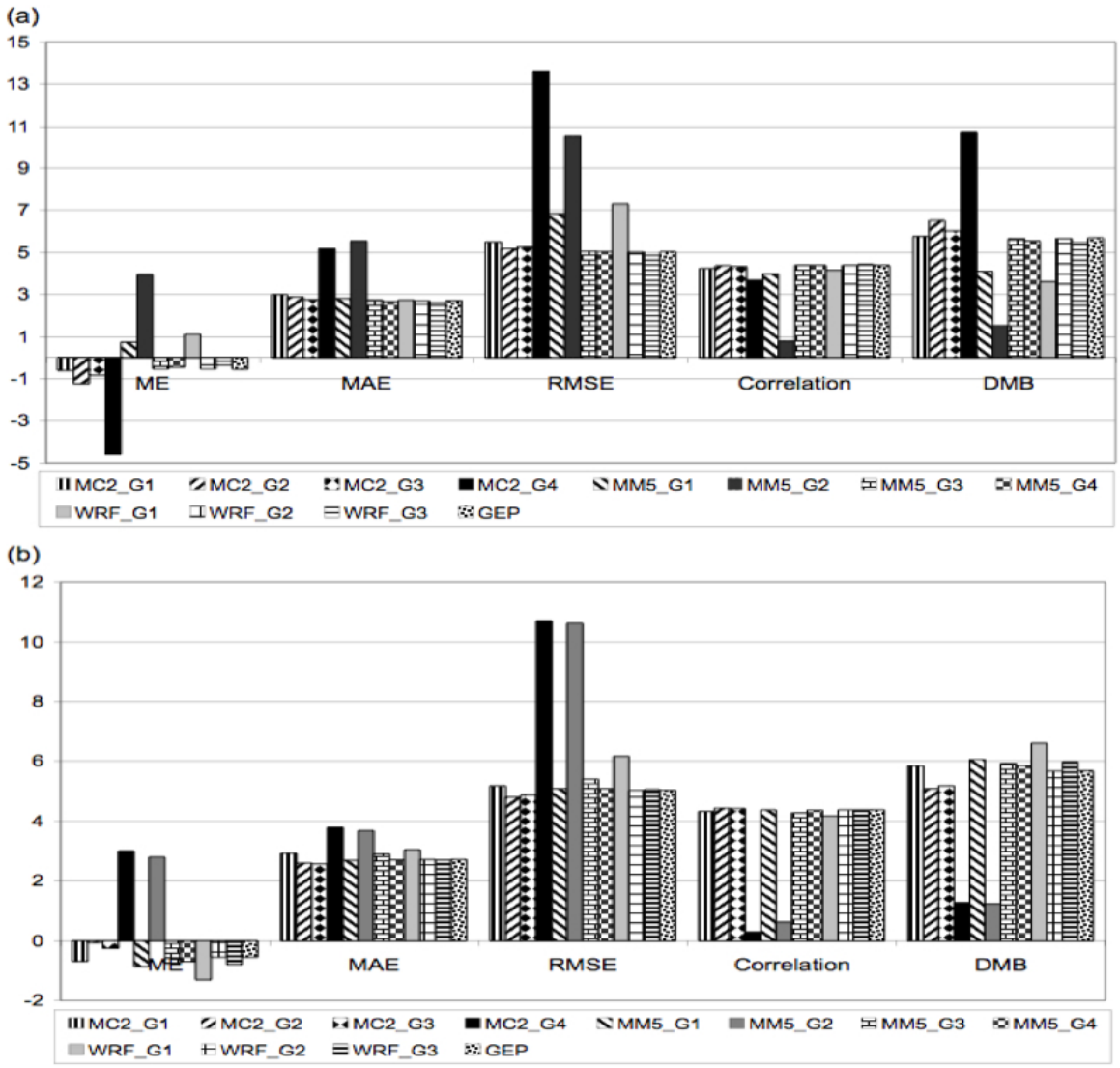


Figure 2.12: Sensitivity study for Vancouver Airport, based on (a) halving the input signals to each GEP ensemble member separately, and (b) doubling the inputs. Error statistics, scaling, and interpretation is as for Figure 2.6. All ensemble members show nearly the same skill as the original GEP except for WRF-G2, MM5-G2, MM5-G1 and MC2-G4, which have significant increases in their error. These "important" ensemble members dominate the outcome of the whole DEF.

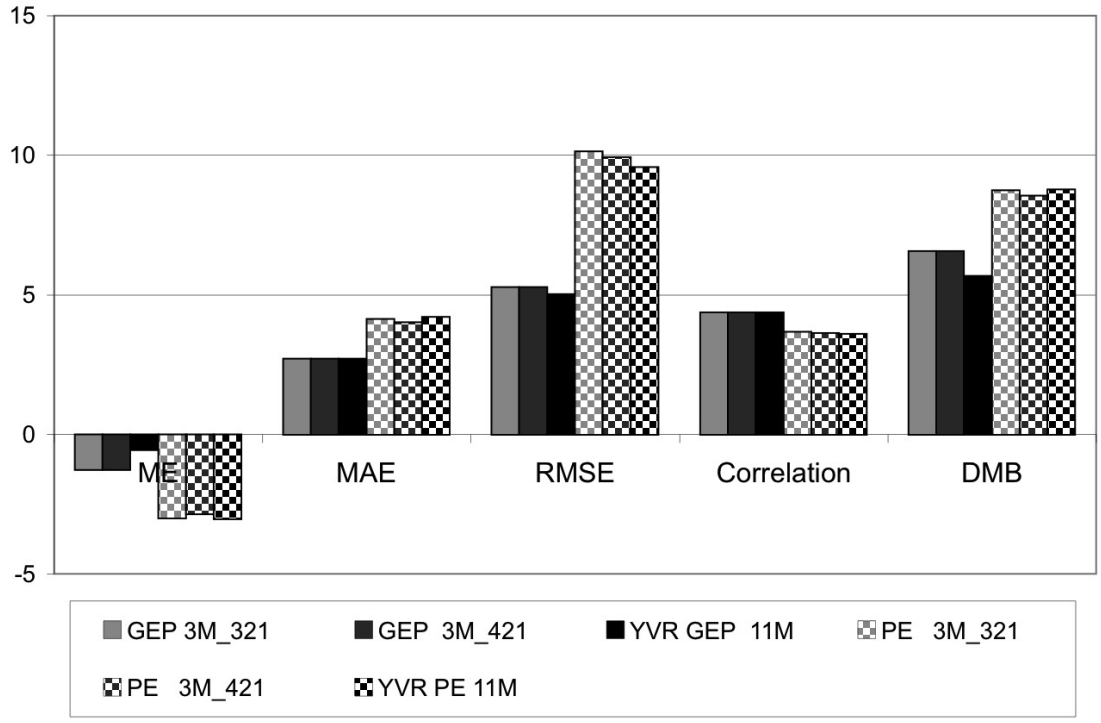


Figure 2.13: Verification statistics for the GEP bias-corrected ensembles at Vancouver Airport for the six different ensemble runs. From left to right (light shading to dark) represent: (1) GEP with 3 members: MC2-G3; MM5-G2 and WRF-G1; (2) GEP with 3 members: MC2-G4; MM5-G2 and WRF-G1; (3) the original GEP with all 11 members; (4) the pooled ensemble with 3 members: MC2-G3; MM5-G2 and WRF-G1; (5) the pooled ensemble with 3 members: MC2-G4; MM5-G2 and WRF-G; and (6) the pooled ensemble with all 11 members. Error statistics, scaling, and interpretation is as for Figure 2.6.

	FINAL SETTING	SETTING I	SETTING II	SETTING III
Population size	40	30	30	30
Number of training sample	107	107	107	107
Number of testing sample	51	51	51	51
Number of scoring samples	41	41	41	41
Number of Functions	80	13	13	13
Terminal sets	11	11	11	11
Head length	15	10	10	10
Number of genes	7	4	**4**	**2**
Linking function between genes	Addition	Addition	Addition	Addition
Mutation rate	0.044	0.044	0.044	0.044
Inversion rate	0.1	0.1	0.1	0.1
IS transposition rate	0.1	0.1	0.1	0.1
RIS transposition rate	0.1	0.1	0.1	0.1
Gene transposition rate	0.1	0.1	0.1	0.1
One-point recombination rate	0.3	0.3	0.3	0.3
Two-point recombination rate	0.3	0.3	0.3	0.3
Gene recombination rate	0.1	0.1	0.1	0.1
Fitness function	RRSE	**RAE**	**RRSE**	RRSE

Table 2.3: GEP parameter settings. RRSE (Root Relative Absolute Error) and RAE (Relative Absolute Error) are fitness functions (see Appendix B). See Appendix C for descriptions of the various genetic modification rates. Differences among Settings I to III are highlighted in bold and underlined.

(a) YVR	**Horizontal Grid Size (km)**			
Model	**108 (Grid 1)**	**36 (Grid 2)**	**12 (Grid 3)**	**4 (Grid 4)**
MC2	*1*	*2*	*3*	**4**
MM5	**5**	**6**	*7*	*8*
WRF	**9**	*10*	*11*	*n/a*

(b) YQQ	**Horizontal Grid Size (km)**			
Model	**108 (Grid 1)**	**36 (Grid 2)**	**12 (Grid 3)**	**4 (Grid 4)**
MC2	*1*	*2*	*3*	**4**
MM5	**5**	*6*	*7*	**8**
WRF	*9*	*10*	*11*	*n/a*

Table 2.4: A map of the "ensemble space" for our case study, showing the 11 members of this multi-model, multi-grid ensemble. Those ensemble members that are most important to the deterministic ensemble forecast appear bold, underlined, and larger. Less important members are indicated with a small italic font. (a) For Vancouver Airport (station YVR). The MM5(Grid 1) was moderately important, and appears bold and medium size. (b) For Comox (station YQQ).

Chapter 3

Electric Load Forecasting using Gene Expression Programming

We introduce a method called gene-expression programming (GEP, a variant of genetic programming) to estimate electric load via a nonlinear combination of weather, calendar and load data. From a population of competing and evolving algorithms (each of which uses a different combination of inputs and functions), GEP uses a computational version of natural selection to find the algorithm that maximizes a verification fitness function for electric load.

For western Canada, the typical electric load curve has relative minima during nighttime and mid-day, and relative maxima in the morning and evening. We estimate the load for each of these four extrema by first removing the trend, annual cycle, and semi-annual cycle. Then, we use GEP to fit the remaining load signal (separately for each extreme) as a function of air temperature, wind speed, precipitation, humidity, day-of-the-week, and other meteorological and calendar variables. This is done via a multi-year training set of load observations and weather data.

Once these best-fit algorithms are found, we use them to forecast load extrema for subsequent days. We then use a Bézier curve to interpolate between the extrema to get hourly load forecasts. This method is verified against independent data for a year of daily load forecasts.

3.1 Electric load and the weather

Electric utility companies plan ahead to match supply with demand, to prevent shortfalls and manage surpluses of energy. One factor that strongly influences electricity demand is weather. Electricity consumption (known as load) is sensitive to air temperature, humidity, cloudiness, precipitation, wind, and also depends on time and day of the week (Lemaître, 2010). Thus, electricity-generation companies and energy-marketing planners can benefit from accurate weather and climate information. Roebber (2010) shows an example where a 27% increase in weather-forecast accuracy can save $2 million per year for electricity production in Ohio.

Load forecasts are often divided in to three range horizons based on forecast period: short-term load forecasts are from one hour to a few days; medium-range load forecasts are from a few days to

a year; and long-range load forecasts are for a year or more (Feinberg and Genethliou, 2005). BC Hydro, the largest electric utility in British Columbia, Canada, uses these load forecasts as follows.

Short-term load forecasts (1 hour to a few days) are used to carry out a real-time load-resource balance. This provides BC Hydro with information about which plants/units to operate (optimized based on dispatch price), the amount of available room for imports and/or exports for the real-time marketers, and the planning and scheduling of short-term outages. This hour-ahead plan is sent to Grid Operations (managers of the transmission lines), who confirm that the plan is acceptable based on voltage and stability criteria.

Medium range load forecasts are divided by BC Hydro into two parts. For 24 h ahead to a week ahead, the load forecasts are used primarily to plan longer-term outages and provide the basis (using the projected load-resource balance) for the limits for pre-schedule market activity. Depending on the price expectations, they can be buying or selling in advance for real-time sales. The longer time frame (up to a year ahead) is used to plan the timing of annual maintenance on all their projects. The medium load resource balance also is given to Grid Operations to assess voltage limits, which in turn determines transfer capabilities for transmission limits.

The long-range load forecast (over a year ahead) is carried out by BC Hydro's system planning group. It is used primarily for ensuring that domestic load growth requirements are met. This forecast impacts development of new energy projects, co-ordination of long-term energy purchase agreements, planning of long-term maintenance programs, and development of management strategies and policies to deal with load growth or decline as population and industry change. Also, all load-forecast horizons are used to satisfy reliability criteria, which are tied to loads (e.g. spinning and operating reserve requirements).

Feinberg and Genethliou (2005) provide a brief review of different load forecasting methods. Among them, statistical approaches for short-term forecasts have drawn much attention over past decades. Load forecasters have used statistical techniques to forecast the next day's peak load, total daily-integrated load (Park et al., 1991), and hourly load.

Different strategies have been employed for statistical forecasting. One is iterative, where load forecasts or observations from a previous time step are re-used as input to the model for the next step. Another is multi-model forecasting, which uses a separate statistical regression for each of the 24 hours during a day. Last is a single-model multivariate approach that forecasts all 24 hours at once (Hippert et al., 2001). The classification of data to different classes such as weekdays and weekends can further increase the number of models and accuracy of the forecast, but such data subdivision might lead to smaller sample sizes in each class and less-robust forecasts.

Many methods have been proposed for flexible nonlinear statistical modeling such as: polynomial regression (Eubank, 1999), Fourier-series regression (Eubank, 1999; Härdle, 1990), wavelet

smoothing (Donoho and Johnstone, 1995; Donoho et al., 1995), B-splines (Eubank, 1999) tree-based models (Härdle, 1990; Lim et al., 2000; Hand, 1997; Ripley, 1996), multivariate adaptive regression splines (Friedman, 1991), projection pursuit (Friedman and Stuetzle, 1981; Härdle, 1990; Ripley, 1996), Bayesian methods (Dey et al., 1998), group methods of data handling (Farlow, 1984), and artificial neural networks (ANNs) (Lee et al., 1992; Mitchell, 1997; Hsieh, 2009).

As an alternative, we investigate evolutionary programming as a nonlinear regression tool for short-term load forecasts. Evolutionary programming has been available since the 1960s: first in the form of "genetic algorithms" (Holland, 1975), and later as "genetic programming" (Cramer, 1985; Koza, 1992). A third, relatively new, variant is called "gene expression programming", was pioneered by Ferreira (2006). Recent studies (Bakhshaii and Stull, 2009; Roebber, 2010) discussed this method with some examples for weather and load forecasting.

We will compare gene-expression programming load forecasts to forecast models with ANNs. ANNs are applicable for most situations in which a relationship between predictors (independents; inputs) and predictand (dependent; output) exists – even when the relationship is very complex. Most neural networks that can learn to generalize effectively from noisy data are effectively identical to other statistical methods. There is considerable overlap between the other statistic methods and ANNs, however advantages for using ANNs include its applicability to fairly large and noisy data sets and fast prediction ability.

Section 3.2 reviews concepts of gene expression programming. The method and data are discussed in Section 3.3 and Section 3.4. Section 3.5 presents the results of the forecast experiment, and compares load regressions (load vs. weather) created using gene expression programming and created using ANN. Conclusions and recommendations are in Section 3.6.

3.2 Gene expression programming (GEP)

Most of the function-fitting approaches mentioned above assume that a functional form (such as a polynomial or an ANN) was chosen a-priori by the user. The regression or error-minimization algorithms are used only to find the parameters or weights in those functions. But how do you know that you picked the best function to start with?

3.2.1 Computational natural selection

Gene expression programming (GEP) assumes that neither the functional form nor the weights/ parameters are known. Instead, it uses a computational version of natural selection to test a wide variety of functional forms and weights until it finds a relatively good one. To do this, it first creates a "world" with a somewhat random population of different candidate functions and weights, and

then it evaluates each candidate against the "training set" of data to find which ones give the best verification scores. These scores define the fitness of each candidate function, which is used to select which ones survive into the next generation. This new generation of candidate functions is again tested against the training data set, and the natural selection process is invoked again. After many generations, the verification score plateaus at some good value – corresponding to a winner relative to the other individuals in that world.

But perhaps a different world, started from a different random set of candidate functions, will follow a different evolutionary path leading to a verification plateau with a relative winner that is even better. To increase the likelihood of finding an absolute winner, many worlds of evolutions are started from different random starting points, and all the relative winners are saved. A separate "testing set" of data (independent from the training set) is used to calculate verification scores for all the relative winners, and is used to avoid over-fitting of the data and to find the overall winner. Ferreira (2006) thoroughly examines how the number of multiple worlds relates to the likelihood of finding an acceptably good answer.

A third independent subset of data, called the "scoring set", is then used to evaluate the final verification statistics for the one winning function. Although the requirement for three subsets of data (training, testing, and scoring) means that a large total data set is needed, it also ensures that the final verification statistics that we present in this paper are truly independent of all data that was used to find a best load-vs.-weather algorithm.

3.2.2 Gene expression programming concepts

By analogy with biology, a candidate electric-load-estimating algorithm is called an individual. The genetic information (i.e., the genome) for an individual is encoded in a chromosome. The chromosome can consist of one or more genes. Each gene is a linear string of ASCII characters, and each gene is divided into a head and a tail. In the head are ASCII characters representing basic functions (such as $+, -, *, /$, ln, sin, conditional tests, etc.), predictors (e.g., meteorological variables), and numbers (e.g., weights). In the tail are only predictors and numbers. Our work uses chromosomes containing five genes, where each gene has a head size of 10 characters.

For example, here is a monogenic chromosome that encodes an algorithm for Planck's Law for emitted radiance: $/a*P-\lambda 5e1/*b\lambda T...$, where the input variables are wavelength λ and temperature T, the basic functions are $(*, /, -)$, the other functions are P representing the power function x^y and e representing $\exp(x)$. Also, a and b are the Planck constants, and ... represents unread characters in the gene tail. By knowing the arity (i.e., the number of arguments that each basic function can have), any chromosome can be used to define an expression tree. This expression tree prescribes the computational form (i.e., the phenome) of the algorithm. In our Planck's Law

example, the algorithm gives the radiance as: $a\{\lambda^5[\exp(b/(\lambda T))-1]\}^{-1}$. Details of the line-by-line (read-like-a-book) procedure to get from the chromosome to the algorithm are illustrated in Bakhshaii and Stull (2009) and are explained in detail by Ferreira (2006).

A set of 50 individuals forms the population of candidate algorithms. The individuals change from one generation to the next as the population evolves. Such modifications in each genome cause changes in the computational algorithm of load vs. weather. The most effective modification is mutation, which is the replacement of a randomly chosen character in the chromosome with a character representing a randomly chosen basic function, a constant, or a predictor (e.g., weather and calendar variables). Computational evolution is accelerated relative to biological evolution by using a mutation rate of 0.044 (e.g., 4.4 mutations per 100-character chromosome per generation). Modification rates are based on Ferreira's (2006) recommendations.

Mimicking biology, the following additional modification methods are implemented in GEP. Asexual modifications include:

- inversion (reversing the character order in a substring, where the start and end positions of the substring are chosen randomly; rate = 0.1, where this means 10% of the population will be modified by inversion during each generation),
- insertion sequence (IS) transposition (a random-length random-location fragment or substring of a gene moves to a different random location in the head of the gene; rate = 0.1),
- root insertion sequence (RIS) transposition (similar to IS, but the substring can move to the front of the gene; rate = 0.1), and
- gene transposition (an entire randomly-chosen gene moves to the front of the chromosome; rate = 0.1).

Sexual (e.g., mating) modifications between two randomly chosen individuals include:

- one-point recombination (one random location in the chromosome is specified, and used to indentify the start of trailing substrings that are swapped between the two individuals; rate = 0.3),
- two-point recombination (the substrings between the same random start and random end point in two chromosomes are swapped; rate = 0.3), and
- gene recombination (randomly selected entire genes are swapped between two individuals; rate = 0.1).

After the individuals in the population have changed from one generation to the next, the training data is used to determine the fitness of each individual in the new generation based on their verification scores. Selection is done by computationally mimicking a roulette wheel where the size of each of the 50 segments in the wheel is proportional to the fitness of each individual. The wheel is "spun" as many times as the population size (50), and the winners are retained into the next generation where they again mutate. This roulette-wheel selection favors the fittest individuals, but still retains some less-fit individuals to help maintain genetic diversity.

The main advantage of gene expression programming (GEP) over the previous evolutionary methods is that GEP is extremely efficient at creating and testing viable candidate functions. This allows the evolution to converge quickly to a solution, and requires no more computer power than a desktop PC.

That is not to say that GEP is totally objective. GEP creates its complex functional forms by drawing randomly from a bag of basic functions and operators, such as $+, -, *, /$, exp, ln, sin, conditional tests, etc. The user of GEP is free to decide what functions are in this bag, and also free to decide the maximum complexity (i.e., the max count of basic functions and weights allowed in the final algorithm). Consider the following trade-offs when selecting the choice of basic functions. If you choose a small number of basic functions, then you might need to allow a longer chromosome (i.e., allow GEP to combine more of these basic functions together) to achieve a fit to the data with the desired accuracy. Alternately, by allowing GEP to choose from a larger number of more complex functions, the same accuracy might be achieved with a shorter chromosome. The net result is that the complexity is often the same, even though the forms of the results are different.

Roebber (2010) points out that if the data to be fitted is simple enough, then it is beneficial to have a small choice of basic functions and a short chromosome, because the functional form can be more readily interpreted by humans. In Appendix G is a list of the basic functions that we allowed in the bag, and a description of the max complexity that we allowed via a computational "chromosome" size. Other issues such as uniqueness of the evolutionary winner have been discussed by Section 2.6.2.

Finally, the best-fit function found by GEP is often amazingly complex, and looks nothing like a function that would have been manually created by a scientist or engineer. An example of one of these functions is in Appendix H. Next, we discuss how the raw load data was preprocessed to separate out easily predictable periodic portions of the signal, leaving a remainder load signal to be fit by GEP.

3.3 Method

The goal of this exploratory work is to forecast hourly load for the next day, utilizing a numerical weather forecast initialized with an analysis from the afternoon of the current day (at 16 PST = 17 PDT = 00 UTC). We used an additive (Feinberg and Genethliou, 2005; Chen et al., 2001) multimodel approach to forecast four different key points in the daily load graph: two peaks and two valleys (Figure 3.1). The advantage of using four models instead of 24 is the flexibility for upgrading. Since load consumption is subject to changes due to events such as shutdowns of large factories and economic growth, frequent upgrading of statistical load models is often unavoidable. A disadvantage is that hour-to-hour load fluctuations that do not follow the smoothed four-extremes daily load approximation are not well forecasted.

Any number of key load points per day could have been picked. Some utility districts have just one morning minimum and one evening maximum. Other electric utilities need six or more daily "turning points" to fit their load curve. However, for our first experiments with GEP, we test a simpler approach with just four turning points, which is satisfactory for many (but not all) days in the BC Hydro utility region.

The model creation procedure has the following steps, which are done on the "training" subset of data (see Section 3.4 for data description):

a) Identify load magnitudes at each of the four extremes (key points identified as C1, C2, C3, C4, as in Figure 3.1) every day.

b) Form separate time series for each key point over the training period (Figure 3.2a)

c) Separately for each key point, regress and then remove the multi-year linear trend (if any), annual cycle, and semi-annual cycle (Figure 3.2b) from the time series, leaving quasi-random-looking residual signals (Figure 3.2c).

d) Separately for each key point, input the residual signal into GEP as the predictand, and input corresponding observed weather variables, previous loads, and day flags as predictors. Run GEP in function-fitting mode (not in time-series mode) to find the best algorithms that approximate the residual as a function of the predictors. Then use an independent "testing" subset of residual data to select the one best algorithm for that key point. Forecasting the load for any one key point can be done as follows:

e) Knowing the day of the year for which the forecast is desired, compute the climate signal by adding together the contributions from the trend line, annual cycle, and semi-annual cycle, based on the regression coefficients from (c). Knowing the forecast weather, use the algorithm

that was found by GEP in step (d) to compute the "residual" component, and add that to the climate signal. Once you have done this for each of the four key points in the forecast day, then:

f) Use a cubic Bézier curve with parameterized shape to interpolate to all the other hours of the day. This final result are the hourly forecasts of electric load for all 24h in the day of interest.

Steps (e) to (f) can be done in hind-cast mode using a "scoring" subset of data containing observed weather data, to evaluate the accuracy of the resulting forecast. It can also be done in operational forecast mode using NWP forecasts as input. Details of all these steps are given below.

3.3.1 Identify key points

During winter, the British Columbia daily load graph Figure 3.1 often has two valleys (early morning and mid-day), and two peaks (mid-morning and afternoon/early evening). These four extrema are the key points we forecast directly. The load magnitudes for these key points are found by dividing the hourly load observations each day into four temporal windows, 00 h−07 h, 07 h−11 h, 11 h−16 h, and 16 h−23 h local time Figure 3.1. Minimum loads are found in the first and third windows, and maximum loads in the second and fourth windows.

3.3.2 Form a time series for each key point

For each key point, its daily value from training data set can be collected in a time series (see Figure 3.2a).

3.3.3 Find and remove the trends and periodic signals

Although GEP could be used to try to explain all variations in the load signal, we found by experiment that many genes in the GEP chromosome would have be "used up" to fit the large annual cycle related to change of seasons, leaving fewer genes to try to explain the day-to-day variations. To simplify the analysis we first remove the large seasonal variations as follows.

It is convenient, but not necessary, to create a normalized time variable t_Y that ranges from 0 to 1 during the course of one year. Namely, $t_Y = d/365$ for a non-leap year, where d is day of the year. For example, 15 February is the 46^{th} day of the year, giving $t_Y = 0.126$.

Standard linear-regression methods can be used to fit a straight line to the multi-year load data for any one critical point. For example, for the afternoon load peak (labeled as key point C4):

$$L_{trend-C4} = a_{C4} + b_{C4} \,.\, t_Y \qquad 3.1$$

Where a is the intercept and b is the slope found by regression. When doing this, it is important to use an integer number of annual cycles (e.g., 3 full years, not 3.7 years), otherwise the small trend will be biased by the large annual cycle. Remove this trend from the original load signal L_{C4} to yield a deviation signal L'_{C4}:

$$L'_{C4} = L_{C4} - L_{trend-C4} \tag{3.2}$$

Next find the amplitudes of the annual cycle via a projection (namely, an inner product) of the load-deviation time series onto cosine and sine waves:

$$A_{1-C4} = 2 \text{ avg } [L'_{C4} \cos(2\pi t_Y)] \tag{3.3}$$

$$A_{2-C4} = 2 \text{ avg } [L'_{C4} \sin(2\pi t_Y)] \tag{3.4}$$

where "avg" is the averaging operator (namely, at each point in the time series, compute the product in square brackets, and then average all these products). A similar harmonic analysis can be done for the semi-annual cycle amplitudes:

$$A_{3-C4} = 2 \text{ avg } [L'_{C4} \cos(4\pi t_Y)] \tag{3.5}$$

$$A_{4-C4} = 2 \text{ avg } [L'_{C4} \sin(4\pi t_Y)] \tag{3.6}$$

You could get the same amplitudes using other methods of harmonic analysis, such as Fourier spectrum analysis or periodogram regression. The reason for including both sine and cosine terms is to capture the phase shift as well as the amplitude in the annual load and semi-annual signals relative to the calendar year. With the above equations, we can reconstruct the "climate" portion of the load signal (see Figure 3.2 b) for any one key point such as C4:

$$L_{clim-C4} = a_{C4} + b_{C4}\, t_Y + A_{1-C4} \cos(2\pi t_Y) + A_{2-C4} \sin(2\pi t_Y) + A_{3-C4} \cos(4\pi t_Y) + A_{4-C4} \sin(4\pi t_Y) \tag{3.7}$$

This "climate" signal includes seasonal variations in load, trends due to slow climate changes (e.g., global warming), and trends due to non-meteorological factors (e.g., population growth, conservation, increased use of energy-efficient appliances). Extrapolating these trends into the future is valid only if no changes in the trend conditions are expected. Important aspects of this "climate" signal are that it is deterministic, does not depend on instantaneous weather conditions, and can be computed in advance for any day of the year.

Finally, subtract this climate signal from the original load time series for that one key point, to

yield a remainder signal Figure 3.2 c:

$$L_{R-C4} = L_{C4} - L_{clim-C4} \tag{3.8}$$

Similar procedures are followed with time series for the other three key points. The remainder signal is what GEP will try to fit as a function of weather and calendar variables.

3.3.4 Use GEP to find the best model

Next, we use GEP to find a functional relationship (i.e., a statistical algorithm or model) between the predictand (load residual) and predictors (weather and calendar variables, see example in Appendix I). This is done separately for each key point. To do this, the data set (14 Aug 2006 to 29 Dec 2009) is split into 3 parts: "training", "testing", and "scoring". For "scoring", almost 30% of the data (recent data) is set aside to use for independent validation of best GEP model. From the remaining of data set, 80% of the dates are randomly selected to form a "training" data set, and the rest are used for "testing" (explained later). A GEP run consists of creating a "world" that contains an initial population of 50 candidate models or individuals, and a statistical evaluation framework to determine the fitness of each individual. Fitness is calculated for each individual at each evolutionary generation, based on both the r^2 value (square of the Pearson product moment correlation coefficient, r) and RRSE (root-relative square error) for the training set. Initially, this statistical score improves rapidly as the models mutate from generation to generation, allowing the fittest to survive. Eventually this error measure reaches a relative minimum, with little further improvement from generation to generation.

RRSE is found as:

$$\text{RRSE} = \left[\frac{\sum_{i=1}^{n}(F_j - O_j)^2}{\sum_{i=1}^{n}(O_j - \bar{O})^2}\right]^{1/2} \tag{3.9}$$

where F is the predicted value, O is the target value, the over-bar represents the mean and the summation is over all events. This fitness function utilizes the total squared error in the numerator, but normalizes it by dividing by a total squared error associated with a simple predictor estimated as the average of all the targets. This fitness function has the desirable trait that it does not narrow the range of individuals too quickly, thereby enabling the wider population diversity that has a greater chance of finding a global-best solution while allowing the evolution to proceed efficiently.

At this point, the GEP run is stopped for this world. Often, several individuals (models with different functional forms) have nearly identical fitness scores. These individuals are saved for future use. Then, a new world is created, again starting from a different random population, and is allowed to evolve to reach its own plateau in fitness scores based on the same "training" subset of

data. Again, the best individuals are saved. After many GEP worlds are run, we end up with a small group of "best" individuals from each of the many worlds.

To select a single model from this group, we now invoke the "testing" subset of data, and evaluate the root mean square error (RMSE), mean absolute error (MAE), and RRSE. This cross-validation allows us to pick the one individual that best predicts load residual, before verifying it against the independent "scoring" data set.

3.3.5 Forecast the daily key load points

To make a load forecast, we rebuild the signal for each key point by adding the residual load (L_R) forecasted by the GEP model to the climate (L_{clim}):

$$L = L_{clim} + L_R \tag{3.10}$$

This is done separately for each key load point. For hindcasts or case studies, observed weather is used as the predictors in the GEP model. For operational forecasts one can use NWP forecasts as the predictors.

3.3.6 Bézier-curve interpolation to get load every hour

We need load forecasts for every hour, but we have load forecasts for only the four key hours each day when load extrema occur. We fit a spline curve to the key points to allow interpolation between those points. To span the full 24 h in a day, we need this curve to start at midnight (earlier than the morning minimum C1), and we need it to extend to the following midnight (later than the afternoon maximum C4). Thus, we need to include an earlier key point C0 (which is just the C4 point from the day before), and a later point C5 (which is the C1 point for the day after), as sketched in Figure 3.1.

A normal cubic spline is not what we need, because it finds the best-fit slope and curvature through each key point. Our key points are at extrema, so we know in advance that the slope must be zero at each key point. Also, given typical shapes of the daily load curve, sometimes the curvature approaching a key point must be different than the curvature leaving. For this reason, we use a cubic Bézier curve, sometimes known as a B-spline. The Bézier spline segment between any two neighboring key points is given by:

$$L(s) = (1-s)^3 L_d + 3(1-s)^2 s L_{dh} + 3(1-s)s^2 L_{ah} + s^3 L_a \tag{3.11}$$

$$t(s) = (1-s)^3 t_d + 3(1-s)^2 s t_{dh} + 3(1-s)s^2 t_{ah} + s^3 t_a \tag{3.12}$$

This is a parametric curve where the load L and time t are functions of parameter s, where $0 \leq s \leq 1$.

At $s = 0$ the curve starts at a departure key point (subscript d) moving toward a departure handle (subscript dh, see Figure 3.3). When $s = 1$ the curve ends at an arrival key point (subscript a) coming from the direction of the arrival handle (ah). For our load problem, the coordinates of the departure key point (L_d, t_d) and arrival key point (L_a, t_a) are known, where the load values are predicted using the GEP algorithm, and the time values are set to the times observed for the extrema on the previous day.

We can simplify the Bézier equations because our key points are at load extrema, where the arrival and departure slopes are zero (Figure 3.3). Thus, $L_{dh} = L_d$, and $L_{ah} = L_a$. To control the shape of the curves, we define time handles based on displacements Δt relative to the key points; namely, $t_d h = t_d + \Delta t_d$, and $t_{ah} = t_a - \Delta t_a$. For our initial work reported here, we use the fixed values of departure displacement Δt_d and arrival displacement Δt_a listed in Table 3.1, which were chosen manually to fit typical load curves reported by BC Hydro for winter electricity consumption. For example, the one spline segment from point C1 to C2 starts using the departure handle underlined first in the center column of Table 3.1 and ends using the arrival handle underlined next. Based on more recent work for other times of year, we recommend that the time intervals Δt be allowed to vary − but more work is needed to find good fits for Δt.

The following steps are used to implement this spline procedure. First, get the key-point load (from Section 3.3.5) and time values and the associated time displacements for the first spline segment from key point C0 to C1. Next, repeatedly solve Eq. 3.12 for time t at parameter value s, where you increment s by some small value (such as $\Delta s = 0.001$) before each solution Figure 3.3. Continue increasing s until time $t \approx$ 1h (within a tolerance set by the utility-company end user), corresponding to the first hour of the day. Knowing that value of s, plug it into Eq. 3.11 to get the corresponding load L at that time, which you save as output. Then, continue increasing s and solving Eq. 3.12. until you get to time $t \approx$ 2h, and use that s to find (Eq. 3.11) and save the corresponding load L. Continue for every hour of the day. If you reach $s = 1$ before you get to your next desired time, then switch to the next pair of key points (C1, C2) and start incrementing from $s = 0$ along this next spline segment. Repeat this process, going from spline segment to spline segment, until you reach the end of your forecast period.

3.4 Data

The case-study area is British Columbia (BC), Canada (Figure 3.4), located between the Pacific coast of North America and the Rocky Mountains. The vast area of BC (almost 950,000 km^2) offers remarkable topographical contrast, characterized by high mountains (2000 to 3000 m), deep narrow valleys, coastlines, fjords, glaciers, ice caps, and interior plateaus. Climate in BC varies from mild marine coastal Mediterranean to hot semi-arid. Köppen climate classes (Peel et al.,

2007) include: Mid-latitude steppe (Bsk), Mid-latitude desert (Bwk), Marine west coast (Cfb), Mediterranean (Csb), Humid Continental (Dfb), Hemiboreal (Dsb), Subarctic (Dfc), Continental subarctic (Dsc).

British Columbia has a population of about 4.5 million people with slightly less than half of this population living in Metro Vancouver. Electricity consumption in southwest BC (called the Lower Mainland and Vancouver Island) represented 70% (as measured by transmission and distribution) to 85% (as measured load distribution) of the BC Hydro system total load during fiscal year 2007-2008 (Wahlgren, 2009), while the remainder of the province was responsible for the rest (see Table 3.2).

Weather and load data were gathered from different sources for this study based on load-distribution information for more than three years (14 Aug 2006 to 29 Dec 2009). The portion of data from 2006 to 2008 was used to train and test GEP, and the portion from 1 January 2009 to the end of the data set was used for an independent verification.

Data from the following four weather stations were selected to represent key climate and population regions: Vancouver, Victoria, Kamloops, and Prince George; representing the Lower Mainland, Vancouver Island, South Interior and Northern Region of BC, respectively (Figure 3.4). BC Hydro provided hourly electric load consumption summed over the whole BC Hydro service area.

We considered a variety of weather, calendar, and past-local variables, as have been tested as predictors in previous studies (Khotanzad et al., 1998; Hippert et al., 2001; Feinberg et al., 2003; Lemaître, 2010). Also affecting our choice of predictors was the availability of variables from NWP outputs for the various geographic sub-domains of BC. As a result, the number of predictors can differ for each key point. Because we train GEP in perfect-prog mode using weather observations instead of NWP-output, the predictors we use are temperature, dew-point temperature, humidity, wind speed and precipitation as provided by Environment Canada, the UBC Emergency Weather Net, and BC Hydro. Because this research will be used for operational load forecasts, the uncertainty of forecast parameters may be an issue. For this reason, cloud darkness is excluded from our list of predictors although it has been recognized as an important variable for load forecasts.

In addition to weather and load data, calendar information is also used as predictors. The month and day of the week are encoded separately. A complete list of variables is provided in Appendix I.

3.5 Forecast test results

3.5.1 Load forecasts

Figure 3.5 shows GEP forecast versus observed total load for the key load points for the independent verification (scoring) portion of data set. Figure 3.6 show forecast versus observed residual load;

namely, not including the deterministic climate signal (L_{clim}). When the key-point forecasts for total load are used with the Bézier interpolation, the result is a forecast of hourly load as illustrated by the four-day sample in Figure 3.7.

3.5.2 Verification statistics

Because each key point has its own GEP-evolved algorithm, each key point is verified independently. First we verify the load amplitude of the key points and then their timed amplitudes, to identify different sources of error. The case study has been done in perfect-prog mode, using weather observations as input instead of NWP forecasts. The verification scores for each key point have been calculated using the 2009 portion the data set (i.e., independent scoring data). Table 3.3 shows verification scores for each critical point.

Two verification methodologies are used: 1) examining only the magnitude of the load extrema (Table 3.3a); and 2) examining the forecast load at the one hour for which that extreme is expected, namely at the same hour as in the day before when the extreme load was observed (Table 3.3b). For example, for the first methodology the first key-point forecast is compared with minimum observed load within the broad morning time window (Figure 3.1) that same day. For the second methodology, assume the key load as forecast by GEP happens at same time as it happened the day before. Table 3.3b presents scores of verification with this timing.

Since the actual load of each key point is highly correlated with load at the previous point, updating the forecast for each key point with observed data from the previous key point can significantly improve the result, as was shown in Table 3.3a and Table 3.3b. Namely, the previous key point load was already included as a predictor (see Appendix I) for the verification values reported in Table 3.3. For comparison, if previous observed load is not used as a predictor, then the verification skill deteriorates as the daily forecast progresses from C1 to C4 (Table 3.4).

Figure 3.8 illustrates the error distribution for each key point while including observed-load updating of past key points in the forecast of future key. Figure 3.8a shows forecast error distribution disregarding the timing of key points, and Figure 3.8b presents the same information but with regard to time of event. The errors at the key points have an almost-Gaussian distribution centered on a positive bias. When a Bézier curve is used to interpolate from the key points to every hour, the resulting hourly forecast error shows a broader and more skewed distribution (Figure 3.9). This suggests that we should also try to predict the time deviations of the Bézier handles to allow a more accurate interpolation.

3.5.3 Comparison with ANN

Compared here are load-forecast results made by GEP and by an artificial neural network (ANN) for the 2009 verification (scoring) year. The particular ANN Short-Term Load Forecaster (ANNSTLF) is a proprietary package (Khotanzad et al., 1998; EPRI, 2009) used operationally by BC Hydro − they provided a file of their historical ANNSTLF forecasts to us. For both GEP and ANNSTLF, the load forecasts are made using weather observations as predictors rather than using NWP forecasts as predictors, to allow us to focus on the abilities of the load algorithms.

Table 3.5 lists forecast errors for the whole 2009 scoring year. It also shows scores segregated into winter (defined by BC Hydro as their rainy season) and non-winter. For both morning key points (C1, C2), ANNSTLF was better. For the midday and evening points (C3, C4), GEP was generally better. Figure 3.10 shows that the error distributions of ANNSTLF are similar to those of GEP at each key point. The bias of GEP forecasts at C1 and C2 suggest that further improvements can be made by bias-correction postprocessing − a topic for our future research.

Note that these ANNSTLF forecasts provided by BC Hydro are one-hour ahead forecasts, while the GEP forecasts are longer range (six-hour ahead) forecasts. Although BC Hydro uses ANNSTLF for many forecast horizons (week, 3 days, 1 day, 1 h), they archived and made available to us only the 1 h data. Our future work will add an hourly update to the GEP forecasts based on the observed load of the past hour.

3.6 Conclusions and recommendations

A closer look at Figure 3.8 and Table 3.3 reveals some insights: (1) the early morning minima showed better performance when the time assumption was added. (2) Figure 3.8 also shows that C1 is over-forecasted most of time, while the other key points have better scores (Table 3.3) and a narrower distribution (Figure 3.8) without considering its timing. There appears to be no single common correction for all key points. However our usage of a single pattern of daily electric load behavior with fixed window intervals for all year could have been partly responsible for the load-forecast errors. Although we used a single strategy for all seasons and weekdays, the result is nonetheless promising, and demonstrates the potential of GEP.

The following changes are recommended for future work. A seasonal classification will allow a more precise description of the shape of the daily load graph and their associated temporal windows. Also electricity load consumption appears to be more related to human behavior during weekends compared to weekdays. Different sets of variables for weekends and a longer history of data could possibly improve the forecast.

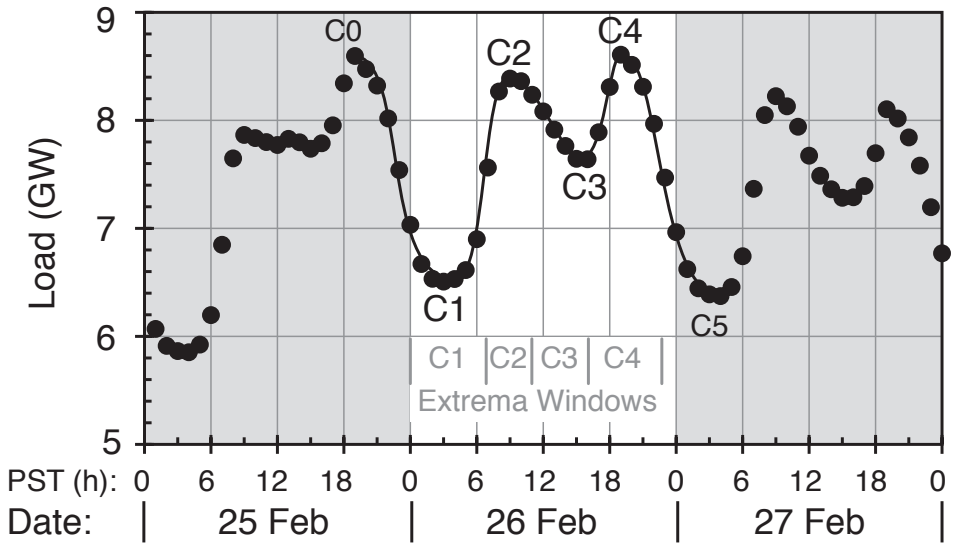


Figure 3.1: Example of observed electric load every hour (data points) during 25 - 27 Feb 2009 for most of British Columbia, Canada, as reported by the BC Hydro Corp. (A small portion of south-central British Columbia is not served by BC Hydro, and is not included in this study.) The typical load signal has two key minima (C1 and C3) and two key maxima (C2 and C4) every day. Together with the max load from the previous day (C0) and min load for the next day (C5), a Bézier spline curve (solid line) can be fit to the data. The timing and magnitudes of the key load points are found within the plotted extrema windows.

Time displacement Δt_a (h) for Arrival toward this key point	**Key Point**	**Time displacement Δt_d (h) for Departure from this key point**
(n/a)	C0 (=C4 from previous day)	3
7	C1	3
2	C2	2
5	C3	2
2	C4	3
7	C5 (= C1 for next day)	(n/a)

Table 3.1: Time displacements for Bézier handles for the spline segments approaching and departing the key points listed in the center column.

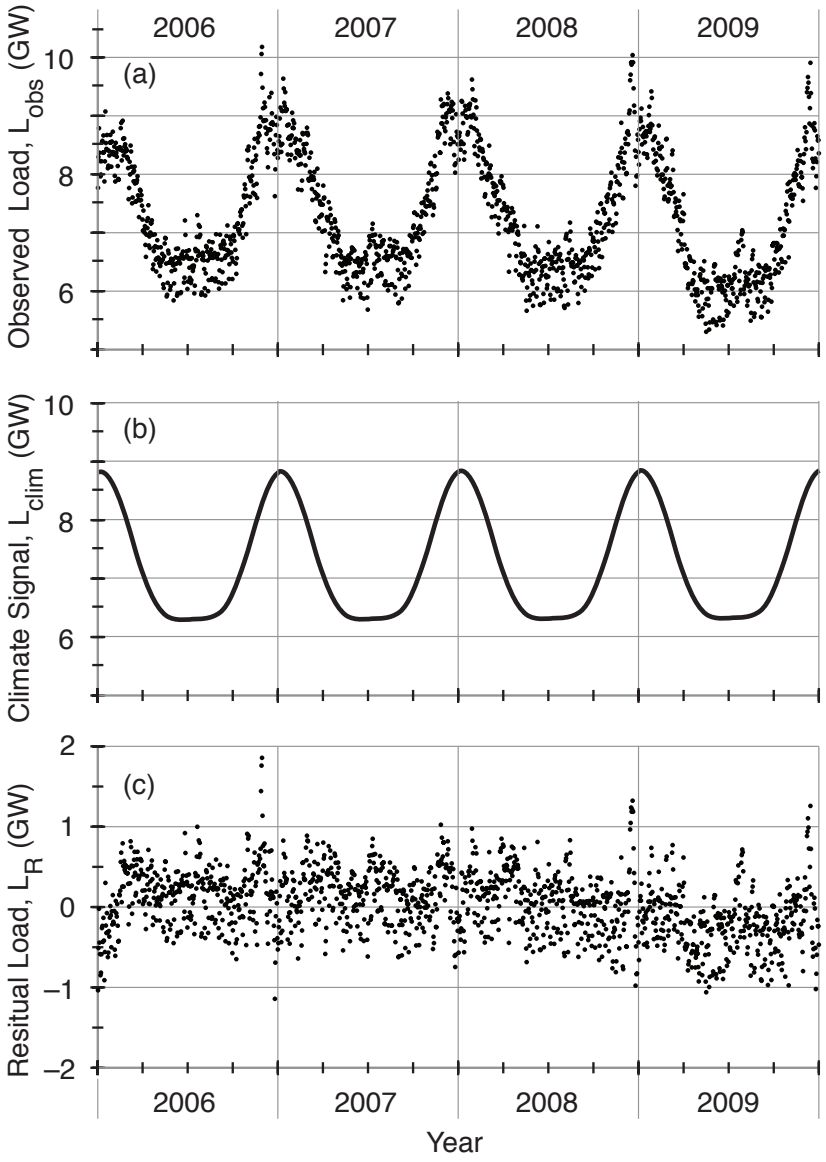


Figure 3.2: Time series of the C4 (late afternoon/early evening) maximum electric load for most of British Columbia, Canada, as served by BC Hydro. (a) Observed C4 load over four years. (b) Best fit "climate" signal for C4 load, consisting of the sum of a linear trend plus regressed annual cycle plus regressed semi-annual cycle. (c) Residual load signal found by subtracting the regressed climate-load signal (b) from the observed load (a). Gene-expression programming is then used to describe this residual as a function of weather and other predictors.

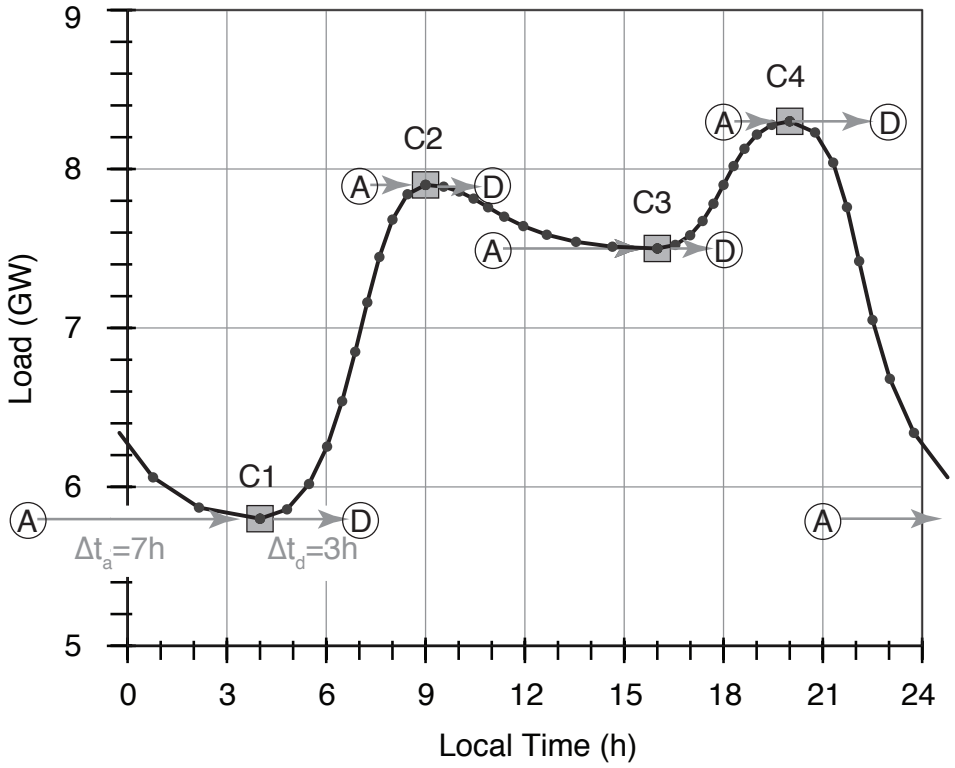


Figure 3.3: Key points (shaded squares) and handles (circled letters) used to describe a Bézier spline curve fit to the electric-load curve for one day. Any one spline segment (such as from key points C1 to C2) departs the initial key point (C1) in the direction of the departure handle (circled D), and smoothly varies to arrive at the next key point (C2) approaching from the direction of the approach handle (circled A). Handles closer in time (Δt) to their key points cause sharper curvature than do handles further away, where Δt_a and Δt_d are arrival and departure time displacements, respectively (Table 3.1). Small dots along each spline segment are for values of the spline parameter s as s varies from 0 to 1 in increments of $\Delta s = 0.1$.

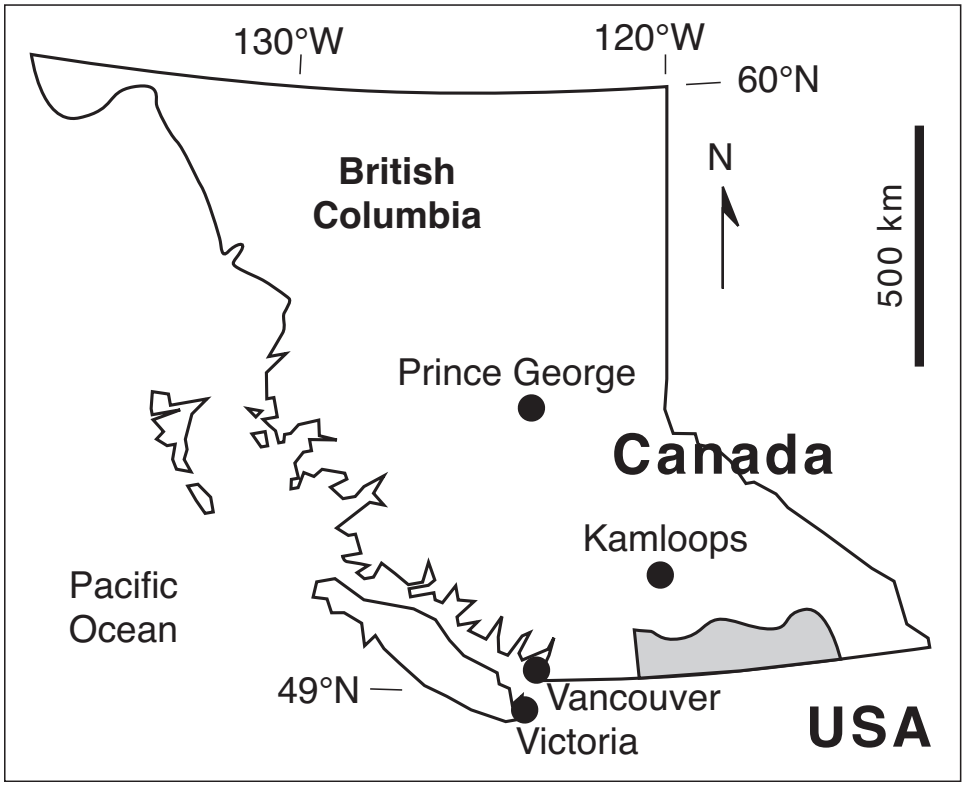


Figure 3.4: BC Hydro service area with corresponding weather stations. The shaded region in south central BC is not served by BC Hydro.

Region Name	Weather Station	Load (GWh)	Proportion of BC (%)
Lower mainland	Vancouver	24,241	47%
Vancouver Island	Victoria	12,038	23%
Northern Region	Prince George	9,828	19%
South Interior	Kamloops	5,672	11%
Total BC		51,779	100%

Table 3.2: BC Hydro system load distribution in British Columbia (2007-2008). Data Source: BC Hydro Database (DLSE).

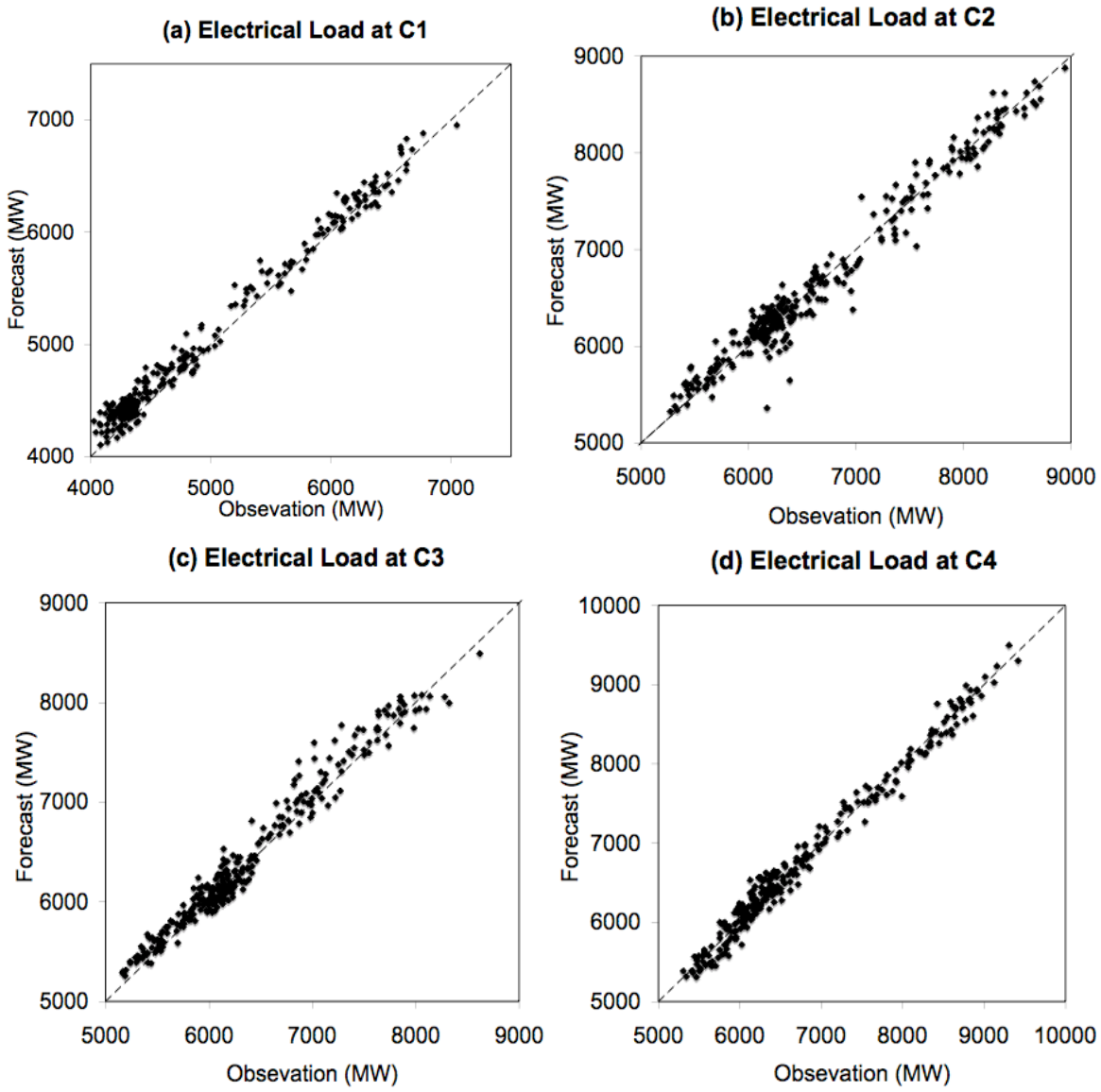


Figure 3.5: Forecast of total electric load in BC vs. observation for each key load point for the independent scoring data set (2009). Corresponding values of r^2 are (a) 0.98, (b) 0.97, (c) 0.97, and (d) 0.98.

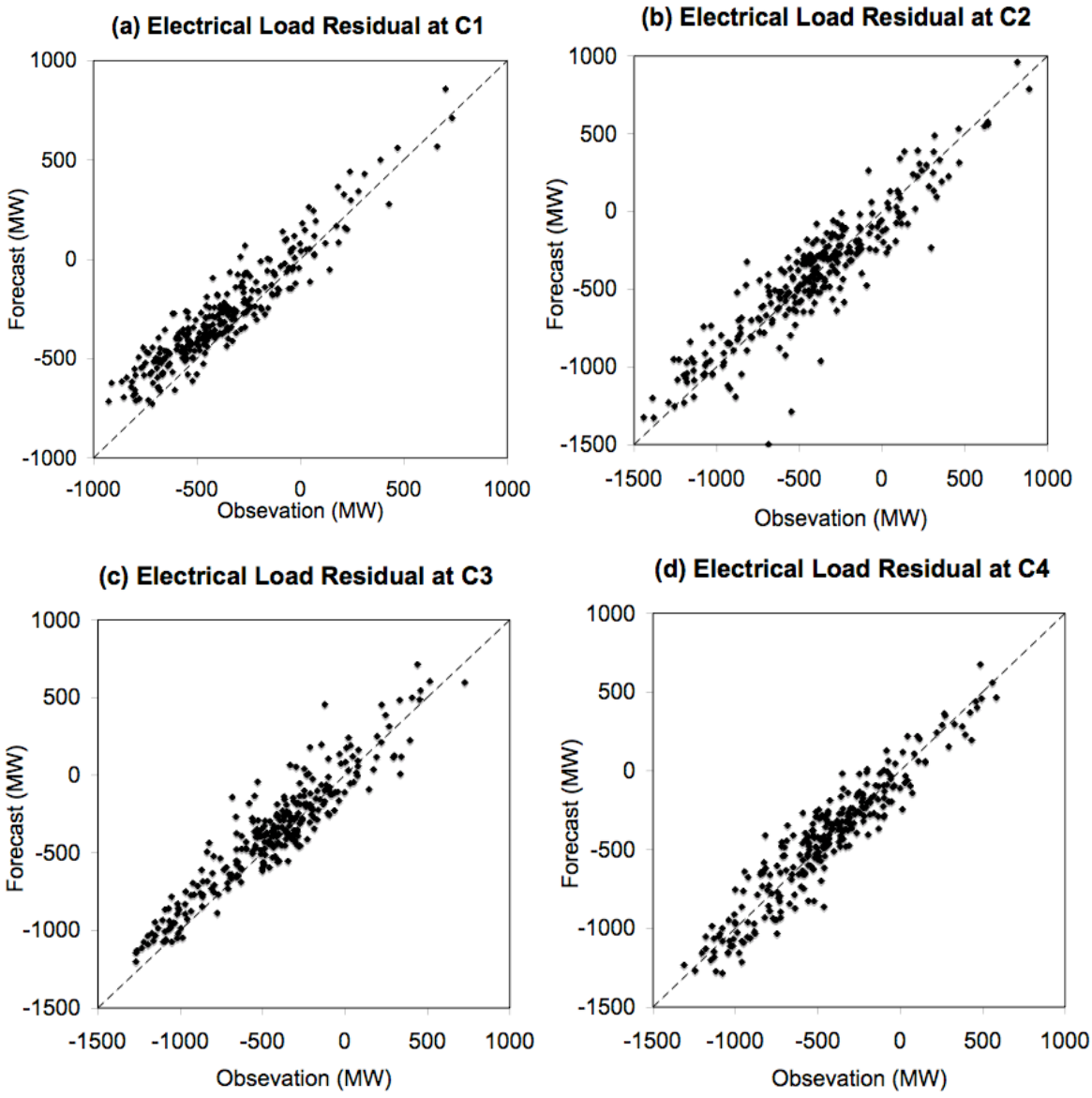


Figure 3.6: . Forecast vs. observed residual portion of the electric load (LR) for the independent scoring data set (2009). Corresponding values of r^2 are (a) 0.88, (b) 0.85, (c) 0.88, and (d) 0.89.

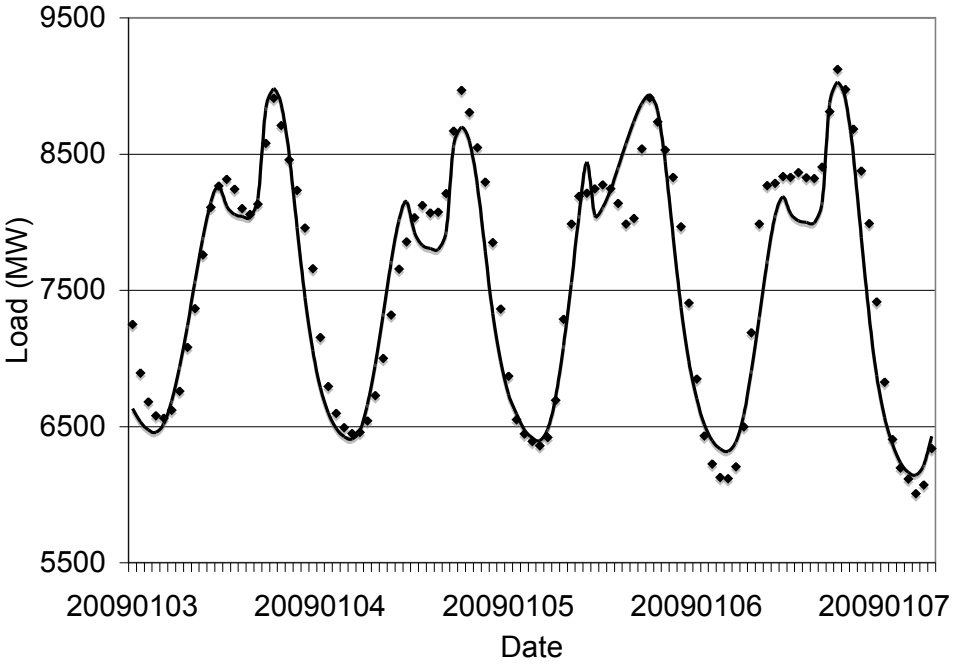


Figure 3.7: Sample of hourly load forecast (solid line) and observations (points) for a four-day segment extracted from the full verification data set. The time code is year (4 digits), month (2 digits), and day (2 digits).

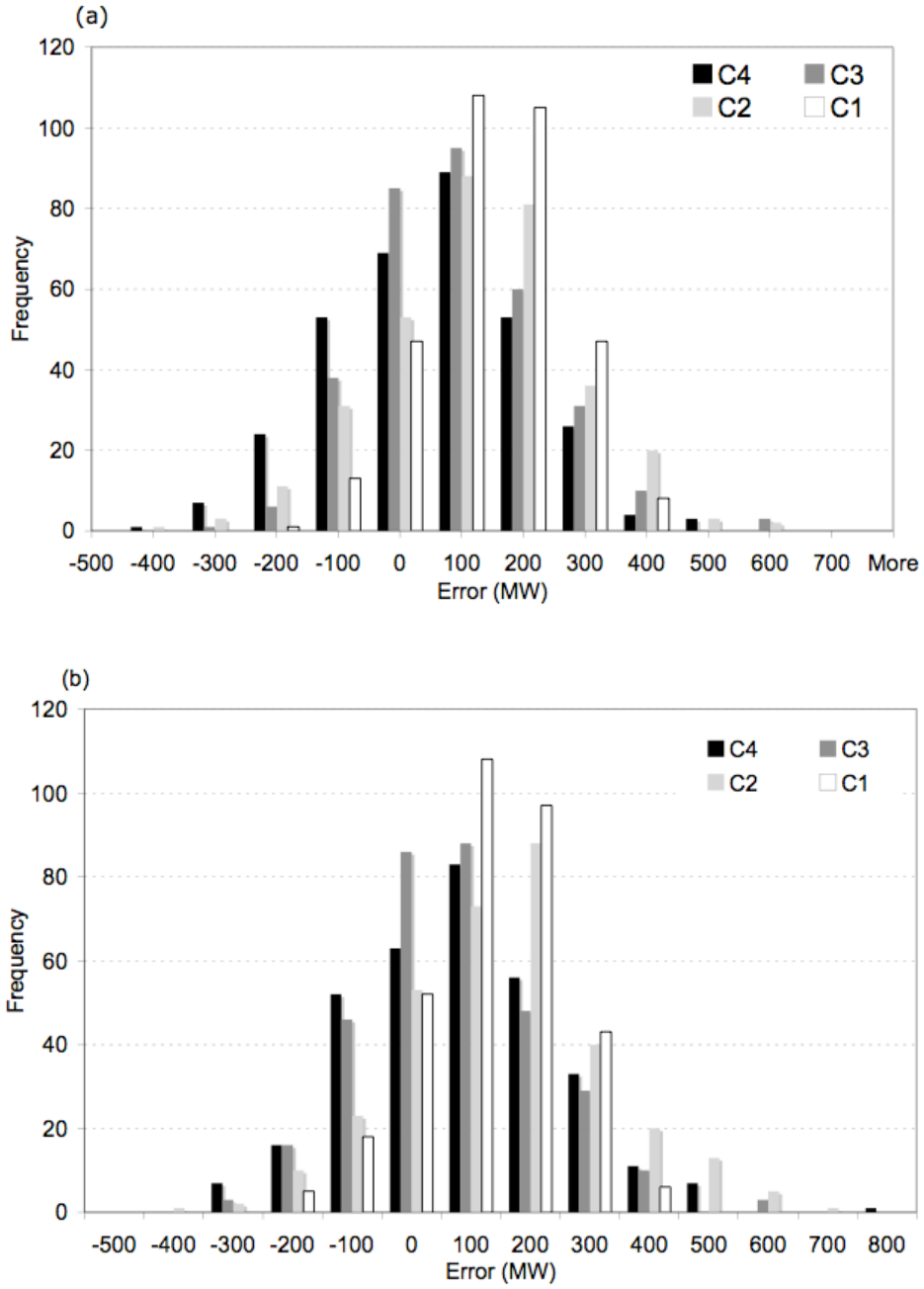


Figure 3.8: The error distribution for each key point from forecasts that include updating past loads with their observations. a) The forecast error distribution disregarding its time. b) Same as (a) but including time.

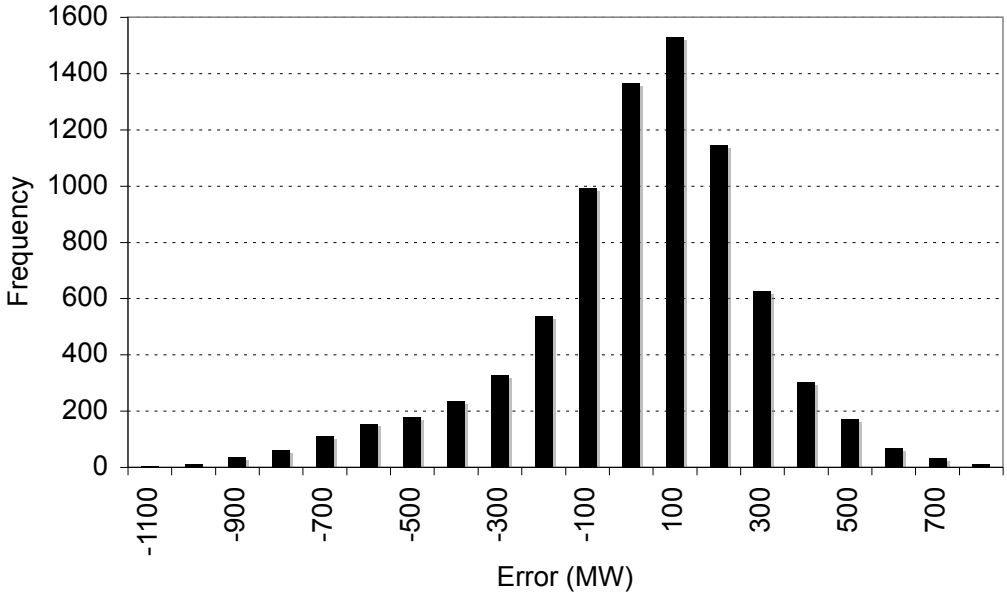
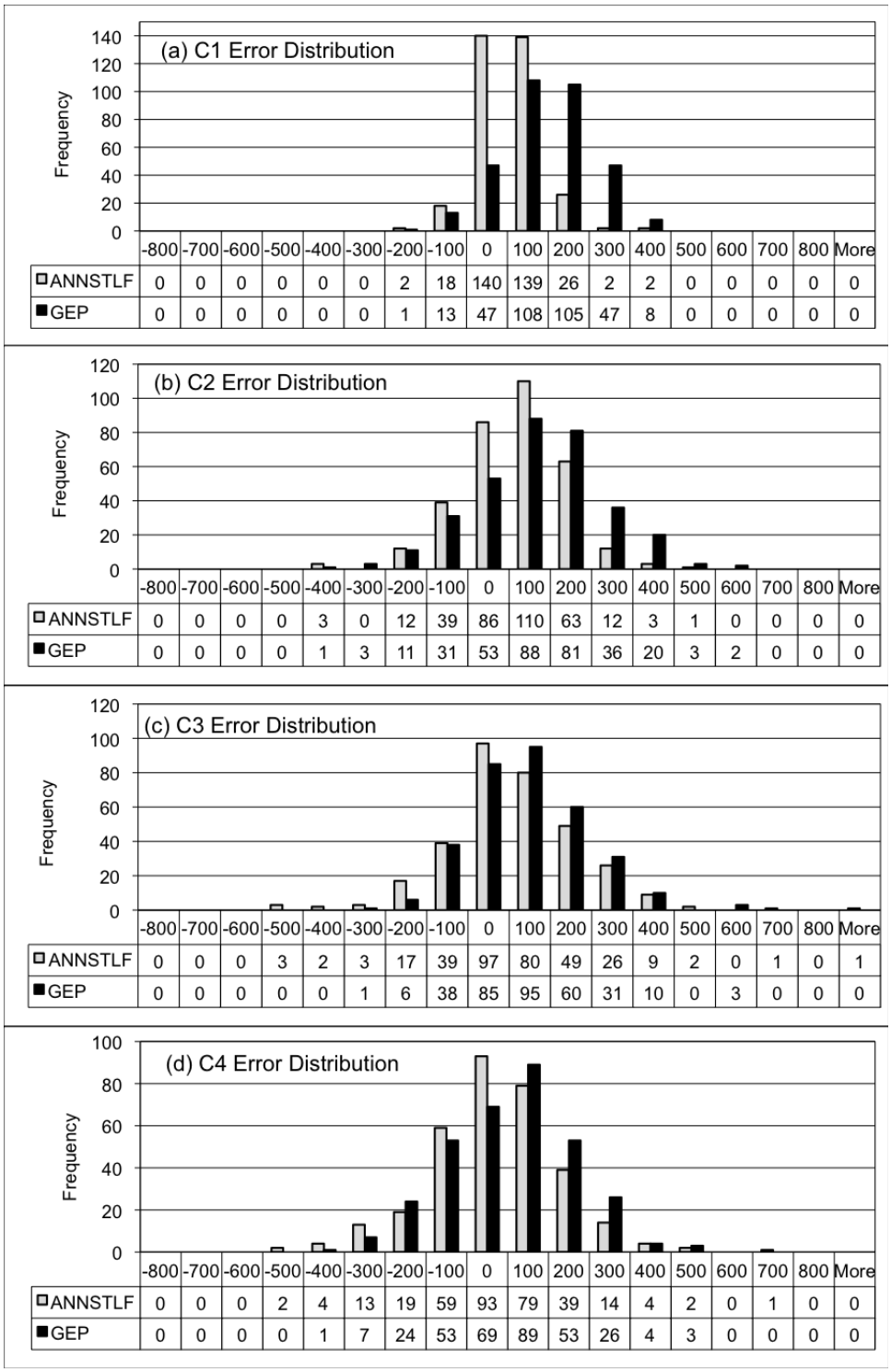


Figure 3.9: Error distribution for the hourly load forecasts.



	-800	-700	-600	-500	-400	-300	-200	-100	0	100	200	300	400	500	600	700	800	More
ANNSTLF	0	0	0	0	0	0	2	18	140	139	26	2	2	0	0	0	0	0
GEP	0	0	0	0	0	0	1	13	47	108	105	47	8	0	0	0	0	0

	-800	-700	-600	-500	-400	-300	-200	-100	0	100	200	300	400	500	600	700	800	More
ANNSTLF	0	0	0	0	3	0	12	39	86	110	63	12	3	1	0	0	0	0
GEP	0	0	0	0	1	3	11	31	53	88	81	36	20	3	2	0	0	0

	-800	-700	-600	-500	-400	-300	-200	-100	0	100	200	300	400	500	600	700	800	More
ANNSTLF	0	0	0	3	2	3	17	39	97	80	49	26	9	2	0	1	0	1
GEP	0	0	0	0	0	1	6	38	85	95	60	31	10	0	3	0	0	0

	-800	-700	-600	-500	-400	-300	-200	-100	0	100	200	300	400	500	600	700	800	More
ANNSTLF	0	0	0	2	4	13	19	59	93	79	39	14	4	2	0	1	0	0
GEP	0	0	0	0	1	7	24	53	69	89	53	26	4	3	0	0	0	0

Figure 3.10: Comparison of load error (Forecast−Observed, MW) distributions for GEP and an artificial neural network (ANNSTLF) for each of the key load points (C1 to C4).

a)	C1	C2	C3	C4
Mean Error (MW)	95.1	68.7	45.9	3.5
Mean Absolute Error (MW)	121.1	136.6	112.6	123.0
Root Mean Square Error (MW)	145.1	172.0	146.2	154.0

b)	C1	C2	C3	C4
Mean Error (MW)	79.8	100.1	24.4	32.6
Mean Absolute Error (MW)	117.9	155.7	118.6	135.8
Root Mean Square Error (MW)	141.5	198.4	152.0	172.7

Table 3.3: Verification scores for the first key point at the early morning minimum (C1), second key point at the morning maximum (C2), third key point at the mid-day minimum (C3) and fourth key point at the afternoon/evening peak (C4): a) Disregarding time, b) Assuming the extrema today happened at the same time as the extrema yesterday, but with the new GEP-forecast load values, and then verifying against the observed load today at that specific time.

a)	C1	C2	C3	C4
Mean Error (MW)	95.1	164.4	183.8	186.9
Mean Absolute Error (MW)	121.1	205.0	231.7	242.6
Root Mean Square Error (MW)	145.1	252.9	281.6	292.2

b)	C1	C2	C3	C4
Mean Error (MW)	80.2	196.1	162.4	215.9
Mean Absolute Error (MW)	118.2	227.5	225.3	266.6
Root Mean Square Error (MW)	141.7	278.8	274.1	324.4

Table 3.4: Verification similar to Table 3.3 but with observed-load updating turned off.

	ANN	ANN_W	ANN_S	GEP	GEP_W	GEP_S
RMSE_C1	73.9	75.9	67.5	145.1	141.5	155.5
MAE_C1	53.6	55.9	46.3	121.1	118.8	128.4
ME_C1	9.5	7.5	15.9	95.1	87.3	118.9
RMSE_C2	125.7	124.9	128.1	172.0	164.3	193.5
MAE_C2	98.0	95.8	104.8	136.6	130.1	156.5
ME_C2	16.1	-4.3	78.4	68.7	56.7	105.2
RMSE_C3	166.7	181.8	107.8	146.2	161.3	84.5
MAE_C3	123.1	135.4	160.5	112.6	127.2	67.7
ME_C3	12.9	6.0	14.8	45.9	53.6	22.1
RMSE_C4	163.5	178.5	104.8	154.0	152.5	158.5
MAE_C4	123.0	135.8	83.9	123.0	123.3	122.1
ME_C4	-28.1	-32.0	-16.3	3.5	32.3	-84.6

Table 3.5: Comparison of verification scores for an artificial neural network (ANN) and GEP for full year 2009, and segregated into a winter (W) rainy season and a non-rainy season (S) . Shown for the four key points (C1, C2, C3, C4) are RMSE is root mean squared error (MW), MAE is mean absolute error (MW), and ME is mean error (MW).

Chapter 4

Electric Load Forecasting for W. Canada: A Comparison of Two Nonlinear Methods

Seven years of hourly temperature and electric load data for British Columbia in western Canada are used to compare two statistical methods, artificial neural networks (ANN) and gene expression programming (GEP), to produce hour-ahead load forecasts. Two linear control methods (persistence; multiple linear regression) are used for verification purposes. A two-stage (predictor-corrector) approach is used, where the first stage uses a single regression model that applies weather and calendar data of the previous hour to predict load for any hour of the day, and the second stage applies different corrections every hour, based on high correlation of today's load with yesterday's load for any given hour. By excluding the day-before variables, the two-stage method reduces the total number of variables in first-stage regression and gives better results. The first five years of data are used for training (regression finding) and validation (comparative testing of candidate functions to reduce overfitting), and the last two for verification (scoring with independent data).

It is found that both nonlinear methods work better than the linear methods for the first stage. All methods work well for the second stage, hence persistence is recommended for second stage because it is easiest. After both stages, the load error is less than 0.6% of the load magnitude for hour-ahead forecasts. When used iteratively to create forecasts up to 24 hours ahead, errors grow to about 3.5% of the load magnitude for both gene-expression programming and artificial neural networks. We also experiment by training the statistical methods with a shorter portion (1 year) of past data to examine the over-fitting problem. Overall, ANN shows better utility in curve fitting on robust data, and GEP is superior on short data sets and is less sensitive to the length of the data set. We also find that a time-nested electric load forecasting model with different lead-times will keep maximum load errors to 2.1% of the load magnitude out to a 24 forecasts horizon.

4.1 Past to present

Electrical load forecasting has been a hot topic in applied meteorology for many years. Mathematical and heuristic techniques, input variable selection, and verification have already been extensively published in literature by other researchers (Alfares and Nazeeruddin, 2002; Hinojosa and Hoese, 2010; Fildes et al., 2008).

This work examines an hourly electrical load forecasting using gene expression programming (GEP) and artificial neural networks (ANN). Genetic programming, which allows algorithms to evolve until they fit a data set, has been used for a wide variety of scientific and engineering applications (Holland, 1975; Cramer, 1985; Koza, 1992). A new variant called gene expression programming (Ferreira, 2006) is very efficient at finding solutions. GEP has been used in load forecasting (Bakhshaii and Stull, 2011; Sadat Hosseini and Gandomi, 2010), numerical weather prediction (Bakhshaii and Stull, 2009; Roebber, 2010; Stull, 2011b), hydrology (Aytek et al., 2008) and many other fields in recent years.

ANN is well established as a tool for electric load forecasting, both as a stand-alone tool (Hinojosa and Hoese, 2010; Musilek et al., 2006; Beccali et al., 2004; da Silva and Moulin, 2000; Hippert et al., 2001; Hippert and Pedreira, 2004; Marin et al., 2002; Mori and Kosemura, 2001; Taylor and Buizza, 2002) and as a hybrid (Liao and Tsao, 2006; Khotanzad et al., 1998; Srinivasan et al., 1999; Yun et al., 2008; Bashir and El-Hawary, 2009; Amjady, 2007; Chen et al., 2004; Fan and Chen, 2006; Ling et al., 2003; Huang and Yang, 2001). While the ANN node-connection diagram is relatively simple (Hsieh, 2009), users might forget that a very large number of free parameters must be determined for ANN. This large number of parameters/weights is both a blessing and a bane. As a blessing, it enables ANN to fit very complex nonlinear behaviors. As a bane, it often means that care must be taken to pick parameters that fit the signal and not the noise (Hinojosa and Hoese, 2010; Ferreira, 2006; Hippert et al., 2001; Osowski and Siwek, 2002; Chan et al., 2006; Mao et al., 2009).

Because gene-expression programming is not widely known in the atmospheric community, we explain the details of it in Section 4.2. Section 4.3 describes the data. Section 4.4 gives the procedure for making one-hour-ahead load forecasts, and that procedure is verified with independent data in Section 4.5. In Section 4.6 we iteratively extend the forecast to 24 hours ahead, and Section 4.7 examines a shorter length of training data. A summary with conclusions is in the last section.

4.2 Gene expression programming (GEP)

4.2.1 Overview of the basic genetic-programming procedure

The procedure is to first create a "world" with a population of randomly created candidate algorithms that relate electric load to predictors such as weather. Next, find the fitness of each candidate by verifying its load forecasts against a "training" data subset. Then select which candidates survive (some with mutation and interchange of genetic information) into the next generation − a selection process that favours the more fit individuals but yet maintains diversity. Verify the new generation of algorithms, and invoke the selection process again. After many generations the best verification score plateaus. The resulting relative winner is saved.

One should repeat this process to create different initial worlds using the same procedure described above, but which can follow different evolutionary paths to different relative winners. It is recommended to use a different subset of data to "validate" and compare each relative winner, to select the one winner that has the best overall fitness statistic with the least overfitting. This validation data is also known as "testing" data by the author of GEP (Ferreira, 2006).

Finally, one should evaluate final statistics of the overall winner using a third data subset, known as "verification" data to the meteorological community, or as "scoring" data in the GEP community (Ferreira, 2006). The three subsets of data (training, validation/testing, and verification/scoring) help reduce over-fitting and ensure independent verification of the final electric-load algorithm. The specific methods used by GEP to achieve this computational selection are described next.

4.2.2 Specific concepts of GEP

By analogy with biology, a candidate electrical-load-estimating algorithm is called an individual. The genetic information (i.e., the genotype, but called the genome in GEP literature) for an individual is encoded in a chromosome. The chromosome can consist of one or more genes. Each gene is a linear string of characters.

Each gene is divided into a head and a tail. In the head are characters representing basic functions ($+, -, *, /$, ln, sin, conditionals, etc.), predictors (e.g., meteorological variables, past load, calendar flags), and numbers (e.g., weights). In the tail are only predictors and numbers. During evolution, no functions or operators are ever allowed into the tail. This artificial division into a head and tail ensures that there will always be sufficient arguments (predictors or weights) for each operator or function in the head to operate on, regardless of the amount of mutation of the head. For this to work, the head length (h) is fixed, and the tail size is computed as $tail = h(n_{max} - 1) + 1$, where n_{max} is the maximum arity. Arity is the number of arguments that an operator or function can take.

As the evolution progresses and individuals with different genotypes are created, these individuals can have different gene lengths. The genes are combined into chromosome using a simple arithmetic operator such as $+, -, *$, or /. We used addition for this study. Figure 4.1 shows an example of how a mathematical algorithm can be represented by a genotype, and how mutation in this genotype gives a new algorithm. What makes GEP different from previous genetic programming methods is that the genome is coded by reading the expression tree like a book, rather than by following the node connections. Decoding back to an algorithm is possible because the arity of each operator and each basic function are known (Ferreira, 2006; Bakhshaii and Stull, 2009). The read-like-a-book coding is the key to the efficiency of GEP, because every mutation gives a viable individual (i.e., a math expression that can be evaluated).

Mutation (change of a randomly chosen character in the gene) is the most effective modification of the chromosome. Computational evolution is accelerated relative to biological evolution (Ferreira, 2006) by using a mutation rate of 0.044 (e.g., 44 mutations per 100-character chromosome per 10 generations). Mimicking biology, the following additional modification methods are also randomly invoked in GEP.

- Inversion: reversing the character order in a random substring; rate = 0.1.
- Insertion sequence transposition: a random substring moves to a different random location; rate = 0.1.
- Root insertion sequence transposition: a random substring moves to the front of the gene; rate = 0.1.
- Gene transposition: an entire randomly-chosen gene moves to the front of the chromosome; rate = 0.1.
- One-point recombination: swapping of trailing substrings from identical starting locations between two genes; rate = 0.3.
- Two-point recombination: swapping substrings between the same start and end points in two chromosomes; rate = 0.3.
- Gene recombination: randomly selected entire genes are swapped between two individuals; rate = 0.1.

Selection of survivors is done by computationally mimicking a roulette wheel that has as many segments as the population. However, the size of each of segment is proportional to the fitness of each individual. The wheel is "spun" as many times as the population size and the winners are

retained into the next generation. Thus, the total population count is constant. This roulette-wheel selection favours the fittest individuals, but still retains some less-fit individuals to help maintain genetic diversity. Ferreira (2006) compares roulette-wheel selection with tournament selection and deterministic selection (where selection is proportional to fitness), and recommends roulette-wheel selection because it has high success rate with reduced computer time for complex real-world problems.

The user of GEP is free to decide what basic functions can be used in the gene, and how many basic functions and weights are allowed in the algorithm. Consider the following trade-offs when selecting the choice of basic functions. If you choose a small number of basic functions, then you might need to allow a longer chromosome to achieve a fit to the data with the desired accuracy. Alternately, by allowing GEP to choose from a larger number of more complex functions, the same accuracy might be achieved with a shorter chromosome. Both approaches often yield similar overall complexities. If the data to be fitted is simple enough, then it is beneficial to have a small choice of basic functions and a short chromosome (Roebber, 2010), because the functional form can be more readily interpreted by humans. The Appendix J shows a sample of a load forecast algorithm created by GEP.

4.3 Data

The case-study area is British Columbia (BC), Canada, located between the Pacific coast of North America and the Rocky Mountains. BC spans 950,000 km^2, and is characterized by high mountains (2000 to 3000 m), deep narrow valleys, coastlines, fjords, glaciers, and interior plateaus. Climate in BC varies from mild marine coastal Mediterranean to hot semi-arid.

BC Hydro provides electricity to 94% of the province's 4.5 million people. About 70% to 85% of BC Hydro's electricity is consumed in Metro Vancouver, a city of about two million people in the southwest corner of BC. For this reason, and following BC Hydro's operational practice, we use temperature at Vancouver International Airport (CYVR) as the only weather input for this focused study. Although this is counter intuitive, it gave better load forecasts than using more weather variables such as dew-point temperature, wind speed and other weather conditions.

Hourly temperature and electric load data are used for the case-study period of 1 Jan 2004 through 31 Dec 2010. This 7-year period is divided into training, testing (validation), and scoring (verification) segments. The first 5.25 years of load and weather data (1 Jan 2004 through 31 March 2009) are randomly divided into 80% for training and 20% for testing (validation) portions. GEP uses the testing (validation) data to find the overall winner from multiple worlds of evolution, while ANN uses the testing data to stop the regression when overfitting begins. The remaining data (1 Apr 2009 through 31 Dec 2010) is used as independent data to score (verify) the results. All of the

verifying graphs, tables, and statistical results in Section 4.5 through Section 4.7 are based on this independent portion of data.

As a "perfect prog" experiment, we use ex-post-facto observed temperatures as input to the regressions; hence, we are evaluating only the quality of the load regressions and not the quality of the weather forecasts. Advantages of the perfect prog approach are that it is independent of the source of weather data, so any changes in operational weather forecast models or in humans that provide adjusted weather inputs do not require GEP or ANN regressions to be re-computed. The disadvantage is that the regression does not correct for systematic errors in the weather variables. But this is actually an advantage, because weather forecasts should have their own "model output statistics" (MOS) postprocessing to correct systematic errors, instead of this weather correction being confounded with the electric load regression.

4.4 Procedure for one-hour-ahead load forecasts

4.4.1 Stage 1: Load forecast

To forecast the electric-load $L(t+1)$ predictand for a valid time of one hour in the future, we use the following six predictors as input:

- present load $L(t)$
- present temperature $T(t)$
- forecast temperature $T(t+1)$ at the valid time
- month index $M(t+1)$ at the valid time
- code for day-of-week $D(t+1)$ at the valid time
- hour $H(t+1)$ at the valid time

where month code is $M = \{01,02,...,11,12\}$ for January through December, hour code is $H = \{01,02,...,23,24\}$, and day-of-the-week code $D = \{1,2,...,6,7,9\}$ for {Sun, Mon, ... , Fri, Sat, Holidays}.

As was reported by others (Hinojosa and Hoese, 2010; Khotanzad et al., 1998; Bashir and El-Hawary, 2009), we also find that the best fits are possible by first splitting the data into three categories: weekdays (Monday - Friday), weekends (Saturday and Sunday), and holidays. Separate regressions are done using each statistical method for each category. Day code was not used for holidays, because all holidays share the same code of 9.

For stage one (the predictor stage), we intentionally do not find separate models/regressions for each hour of the day. Instead, we find the regression that gives a 1-hour-ahead forecast, based on a model that was trained using all hours in any one category such as weekdays. Our preliminary experiments showed that even though this approach yielded load forecasts with substantial errors after stage one, these load errors were very highly correlated from one day to the next. Other researchers have used weather and load variables of a day-before forecast day to get around this issue (Taylor and Buizza, 2002; Khotanzad et al., 1998; Drezga and Rahman, 1998). Based on these preliminary experiments, we hypothesize that a two-stage (predictor-corrector) approach can yield accurate forecasts with relatively few free parameters.

We compare two different methods to create the statistical relationship between predictors and predictand: gene expression programming (GEP):

$$L(t+1) = \mathrm{GEP}_1[L(t), T(t), T(t+1), M(t+1), D(t+1), H(t+1)] \tag{4.1}$$

artificial neural network (ANN):

$$L(t+1) = \mathrm{ANN}_1[L(t), T(t), T(t+1), M(t+1), D(t+1), H(t+1)] \tag{4.2}$$

where the subscript 1 denotes the first stage in this procedure. As a baseline we compare the nonlinear-method results against the results from two linear methods: multiple linear regression (MLR):

$$L(t+1) = \mathrm{MLR}_1[L(t), T(t), T(t+1), M(t+1), D(t+1), H(t+1)] \tag{4.3}$$

persistence (PER):

$$L(t+1) = \mathrm{PER}_1[L(t)] \tag{4.4}$$

GEP is programmed to use the following functions and operators: $+, -, *, /$, exp, ln, $()^2, ()^3$, $()^{1/2}$,$()^{1/3}$, sin, cos, arctan. The first four of those operators have an arity (number of arguments) of 2, and the next 9 have an arity of 1. The multigenic chromosome used 5 genes, each gene with a head size of 8 and length of 25 (Ferreira, 2006). The Appendix J illustrates an algorithm produced by GEP.

ANN is programmed as a feed-forward, back propagation network (Taylor and Buizza, 2002) with an input layer of 6 nodes, one hidden layer of 10 nodes, and an output layer of 1 node. This ANN requires that 81 free parameters be determined. We experimented with different counts of nodes in the hidden layer. As the node count increased past 10, the improvement in verification errors plateaued. Hence, we use 10 hidden nodes for this study. ANN uses a hyperbolic tangent sigmoid transfer function $[y = 2/(1+\exp(-2x)) - 1]$ on the input and a linear transfer function on

the output. MLR solves a set of simultaneous equations to give the weights for a least-squares best fit between observed loads and the weighted sum of predictors for the training data. For PER we assume that the load forecast is the same as the present load.

4.4.2 Interim behavior: Motivation for a second stage

Figure 4.2 shows a small sample of the single-stage, one-hour-ahead load forecasts during the training period. All the methods (both nonlinear and linear) capture the first-order signal; namely, the proper order of magnitude (5000 to 10000 MW), and typical variations during the day (e.g., peaks in late morning and early evening). Define a load error E for any hour $(t+1)$ as

$$E_1(t+1) = F_1(t+1) - O(t+1) \tag{4.5}$$

where F is the forecast value from the load-forecast stage, O is the verifying observation at that hour, and subscript 1 still denotes the first stage. All four statistical methods have roughly the same errors: on the order of plus or minus a few hundred MW (see a sample of two methods in Figure 4.3). Of these four methods, persistence has the greatest errors. An interesting pattern is observed in the errors, which points to even further improvement.

When all weekday errors are plotted versus hour of the day (Figure 4.3) for any month in the training period, a recurring error pattern is evident for GEP_1. Namely, the error for any hour is strongly correlated with the errors for that same hour on most other days in that same month for the same category (weekday, weekend). For example, in Figure 4.3a, at 0700 local time in January almost every day has an error in the range of -500 to -300 MW, while at 2000 most errors are in the range of $+200$ to $+400$ MW. Hence, we can use this correlation to reduce errors further, via a bias-correction second stage. To our surprise, incorporation of the day-before load as an additional predictor in stage 1 (not shown here) does not give as good results as the two-stage method. This characteristic is true for both GEP and ANN. So this study focuses on the two-stage method. Note that ANN_1 has substantially lower errors (Figure 4.3b) after stage 1 than the other methods for weekdays and weekends. Also, these errors do not exhibit a strong correlation from one day to the next. This confirms that the more complex model (ANN) with more free parameters (81 free weights for our ANN with 10 nodes in the hidden layer) performs better than the simpler models (GEP has up to 10 free weights, MLR has 6 free weights, and PER has zero free weights in our implementation). We intentionally did not test more complex forms of GEP and MLR with more free weights because we wanted to evaluate the capability of lower-order regressions.

4.4.3 Stage 2: Recurring-bias correction

Figure 4.3 suggests that the error for any hour today should be nearly equal to the error yesterday during that same hour for $\mathrm{ANN}_1, \mathrm{GEP}_1, \mathrm{MLR}_1$, and PER_1. The simplest such correction would use persistence (PER); namely, assume the bias $E(t+1)$ for any hour later today is exactly equal to yesterday's bias $E(t+1-24)$ for the same hour. Alternately, we could use yesterday's bias as one of the predictors in a second regression stage, which would be a seventh predictor. We compare bias corrections using the following statistical methods:

$$E(t+1) = \mathrm{GEP}_2[E(t+1-24), L(t), T(t), T(t+1), M(t+1), D(t+1), H(t+1)] \tag{4.6}$$

$$E(t+1) = \mathrm{ANN}_2[E(t+1-24), L(t), T(t), T(t+1), M(t+1), D(t+1), H(t+1)] \tag{4.7}$$

$$E(t+1) = \mathrm{MLR}_2[E(t+1-24), L(t), T(t), T(t+1), M(t+1), D(t+1), H(t+1)] \tag{4.8}$$

$$E(t+1) = \mathrm{PER}_2[E(t+1-24)] \tag{4.9}$$

where subscript 2 denotes the second stage. So the approach in stage two is that we are finding separate regressions (or using separate persistence values) for each hour.

With 4 statistical methods for stage one and 4 methods for stage two, there are 16 different combinations that could be tested. We tested the following subset of 7 combinations:

- ANN_1-ANN_2
- GEP_1-GEP_2
- MLR_1-MLR_2
- ANN_1-PER_2
- GEP_1-PER_2
- MLR_1-PER_2
- PER_1-PER_2

This subset was chosen because it would likely span the range of outcomes.

Figure 4.4 shows the remaining error after using this two-stage process for the same two-day sample as Figure 4.2. The remaining error is reduced to plus or minus a few tens of MW. Namely, the mean absolute error is of order 0.6% of the total electric load.

To recapitulate, the statistical regressions for stages 1 and 2 were trained and tested using the whole 5.25 years of training data. For stage 1, the predictor stage, each statistical model was trained using every hour of the day. Namely, there was just one MLR model that applied to all hours of weekdays, one MLR for all hours of all weekends, one GEP for all hours of all weekdays, etc. Then for stage 2, the corrector stage, separate regressions were created for each hour of the day for each category (except for holidays), where the number of data samples for training and testing each of these categories (weekdays, weekends, holidays) was 31560, 13104, 1320 respectively. These regressions will be verified against the remaining 1.75 years of independent data, with results given in the next section.

4.5 One-hour-ahead forecast verification using independent data

The best-fit regressions as determined from the training data above were used with no change to predict the load for the independent subset of data. Each of the seven combinations of first and second stages was verified (scored) separately within each category for the independent data set of 1 Apr 2009 through 31 Dec 2010. The number of verification data points for each category (weekdays, weekends, holidays) was (10512, 4344, 480), respectively.

We also compare the results from these 7 combinations with the operational one-hour-ahead load forecasts as provided by BC Hydro. We caution the reader that we had no control over the conditions under which these operational values were determined. BC Hydro runs the commercially available Artificial Neural Network Short Term Load Forecaster (ANNSTLF), which utility companies can obtain from the US Electric Power Research Institute (EPRI, 2009). Although BC Hydro inputs forecast temperatures into ANNSTLF for their real-time operational runs, they later go back and re-run ANNSTLF with the ex-post-facto observed temperatures, and then they archive the resulting load "forecasts". We used these archived (observed-temperature-based) ANNSTLF one-hour-ahead load forecasts as a benchmark, against which we measure the success of the proposed methods.

For the verification statistics, F_i is the forecast value for data point i from any of the statistical methods, V_i is the corresponding verifying observation, and the overbar indicates an average over all N data points. The error is $E_i = F_i - V_i$ and mean error (bias) is $ME = \bar{E} = \frac{1}{N}\sum_{i=1}^{N} E_i = \bar{F} - \bar{V}$. The error standard deviation is $STD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(E-\bar{E})^2}$. The mean absolute error is $MAE = \frac{1}{N}\sum_{i=1}^{N}|E_i|$. The Pearson product-moment correlation coefficient is $r = \frac{1}{N}\sum_{i=1}^{N}[(F-\bar{F})(V-\bar{V})]/(\sigma_F.\sigma_V)$, where

the product of individual standard deviations σ is $\sigma_F.\sigma_V = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(F-\bar{F})^2}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(V-\bar{V})^2}$. Recall that r^2 equals the portion of total variance that is explained by the regression.

Comparison of Table 4.1 and Table 4.3 shows that all 7 combinations of methods are improved by including the second stage. The linear methods MLR_1-MLR_2, MLR_1-PER_2, and PER_1-PER_2 explain between 95.3% and 99.7% of the variance. The methods with a nonlinear first stage and any second stage (GEP_1-GEP_2, ANN_1-ANN_2, ANN_1-PER_2, GEP_1-PER_2) explain between 99.1% and 99.7% of the total variance. For the two-stage methods with a nonlinear first stage, the mean absolute errors are in the range of 41 to 76 MW. Thus, the two-stage methods used here with nonlinear first stage have better verification statistics than BC Hydro's operational version of ANNSTLF (explained variance of 95.6% - 98.9%, and MAE of 81 to 164 MW, for this data set).

The standard-deviation differences in Table 4.1 are highly statistically significant for weekdays and weekends, due to the very large sample sizes (10,512 and 4,344) of the independent data. For example, for weekends a standard-deviation difference of SDT = 1 MW or more between any two methods in Table 4.1 are different from each other at a significance level of better than 0.1% based on a two-sided F test. For holidays, with only 480 data points, a standard-deviation difference of 2.2 MW or more is significant at better than the 1% level.

Next, focus on the models with nonlinear stage 1 with persistence or nonlinear stage 2. Figure 4.5 shows load-error distributions for ANN_1-PER_2 and GEP_1-PER_2, where the figure components a, b, c are for weekdays, weekends, holidays respectively. Figure 4.6a and Figure 4.6b show error distributions for GEP_1-GEP_2 and ANN_1-ANN_2 for weekdays and weekends. Stage-2 has not been applied for holidays because for most holidays do not have a day-before holiday that has same load error pattern for stage-2. For comparison, the operational ANNSTLF results are also plotted. In general, better forecasts are ones with higher central peaks and smaller tails.

Figure 4.5a and Figure 4.5b are very similar to Figure 4.6, suggesting that any reasonable stage two (linear or nonlinear) works well. GEP and ANN have nearly identical error distributions for weekdays, and both are better than ANNSTLF for this data set. For weekends, ANN is best. For holidays, GEP is best.

Another way to compare the two-stage GEP, ANN, MLR with the operational ANNSTLF is to count how many days each method gave the best load forecast. These counts were divided by the total number of days in each category of weekdays, weekends, and holidays to give the relative counts in Figure 4.7. ANN wins for weekends and weekdays, while GEP wins for holidays. The two-stage GEP and ANN give the best forecasts much more often than the operational ANNSTLF.

Both ANN and GEP give excellent results for our data set, when used as the first stage in a two-stage statistical scheme. For holidays GEP is slightly better, perhaps because it has less of an over-fitting problem for small data sets compared to ANNs. For weekdays and weekends, ANN is

better most often. The observation that each of those methods can be best for a substantial portion of days suggests that they could all be combined into an ensemble forecast – a topic for future research.

4.6 Multiple-hour ahead forecasts

The best-fit statistical regressions from the one-hour-ahead forecasts can be applied iteratively to give forecasts more hours ahead. Namely, those equations gave $L(t+1)$ as a function of $L(t), T(t)$, $T(t+1)$, etc. So once we have estimate the load at $t+1$, we can apply that load to find the load at $t+2$ (Taylor and Buizza, 2002; Khotanzad et al., 1998). For example, for GEP stage one: $L(t+2) =$ $\text{GEP}_1[L(t+1)), T(t+1), T(t+2), M(t+2), D(t+2), H(t+2)]$where $T(t+2)$ must be provided as a temperature forecast if used operationally or as ex-post-facto observations if used for perfect-prog research at that valid time. The M, D, and H variables are all calendar variables that are known in advance. Similar iterative equations can be made for the other statistical methods. Stage 2 can be applied similarly.

This process can be iterated more hours into the future. However, by using the forecast load rather than observed load as input on the right side of these equations, the errors will accumulate, and forecast quality will deteriorate with time. We demonstrate this using GEP_1-GEP_2 and ANN_1-ANN_2 forecasts, for which we calculate verification statistics from the independent data set for forecast hours out to 24 h into the future. The result, in Figure 4.8 using observed temperatures as input, shows that while forecast quality does indeed deteriorate with time into the future, the errors are nonetheless small enough to be useful to operational load forecasters.

As an alternative experiment we applied the same two-stage method for a single, large, 24-hour time step. We found that the single-time-step lead-time 24h forecast had error that was 2.1% of total load, compared to the hourly iterative approach that gave error of 3.5% of total load.

4.7 Length of the training data set

In this study we had the advantage of a long, 7-year, data set that we could split into training, testing, and verification subsets. But how important was the length (5.25-years) of the training subset? Could an adequate forecast be made with just one year of training? Which statistical technique is more sensitive to a short training data set?

To answer these questions, we re-ran GEP_1 and ANN_1 to find new best-fit algorithms for stage one using a 1-year training data set (1 Apr 2008 - 31 Mar 2009). For stage 2 we used PER_2 based on that same 1 year of training data. We then verified GEP_1-PER_2 and ANN_1-PER_2 against the 1.75 years of independent data from 1 Apr 2009 through 31 Dec 2010 as was used before. Table 4.2

shows the verification results for the one-hour ahead forecasts.

The short (1-year) training gives verification scores almost as good as the full 5-year training period by applying two-stage forecast. For both weekends and weekdays the short GEP is slightly better than the one trained over a longer time (Table 4.3). This might be an artifact of the evolution procedure, because the shorter data set enables faster evolution, thereby testing a wider range of candidate functions during any given wall-clock time on the computer.

We were curious to see how stage one performed alone, for the shorter data set. The results are in Table 4.3. The artifact of the evolution period is clearly evident in the GEP data. Although it is clear that ANN has the least error for a single-stage load forecast with this short data set, the performance of ANN with a short data set is considerably worse than with a long one. As a result, GEP has better performance than ANN in the second stage for weekdays

4.8 Summary, discussion and conclusions

Using 7 years of weather and electric load data for British Columbia, Canada, we find that a two-stage (predictor-corrector) statistical process gives good and robust load forecasts. Nonlinear statistical methods (gene-expression programming GEP and artificial neural networks ANN) work best for the first stage, and they capture most of the variance in the one-hour-ahead load signal. The remaining errors seem to have a repeating pattern from one day to the next. This allows one to use the error from 24 hours ago as a bias corrector in a second statistical stage to improve the forecast for the present hour. For this second stage, almost any nonlinear or linear method (including 24-hours persistence) works fine. This two-stage method reduces the forecast mean-absolute error to about 0.6% of the total electric load. The hour-ahead forecasts can be used iteratively to forecast more hours ahead, with the error increasing to about 3.5% of the total load after 24 hours of iteration.

To our surprise, incorporation of the day-before load as an additional predictor in stage 1 did not give as good results as the two-stage method. This characteristic was true for both GEP and ANN. So this study focused on the two-stage method.

Using additional weather variables such as dewpoint temperature, wind speed and weather condition did not add value to our one-hour-ahead forecasts (not shown here). However it slightly improved 24 hour lead-time forecasts.

Both GEP and ANN are commercially available as software packages (we used GeneXproTools by GepSoft, and used the ANN package in Matlab) that can run on desktop computers. GEP is a recent variant of genetic programming that evolves much faster and more efficiently. ANN has been proven by others to be able to fit almost any function, but can have a problem of overfitting (i.e., fitting both the signal and the noise) that is less of a problem for GEP. This overfitting difficulty might explain why ANN had a problem fitting the load signal for holidays, for which the data set

was relatively small.

ANN gives the best load forecasts for weekends and weekdays in this case study. It has many more free parameters than GEP, yet GEP takes longer (hours) to reach its best fit via evolution than it takes ANN (minutes) to reach its best fit via back-propagation error minimization. Both methods yield load algorithms that can be solved in seconds for any forecast hour. Future work will include combining load forecasts from GEP for holidays with ANN for weekends and weekdays, with different lead-time nesting.

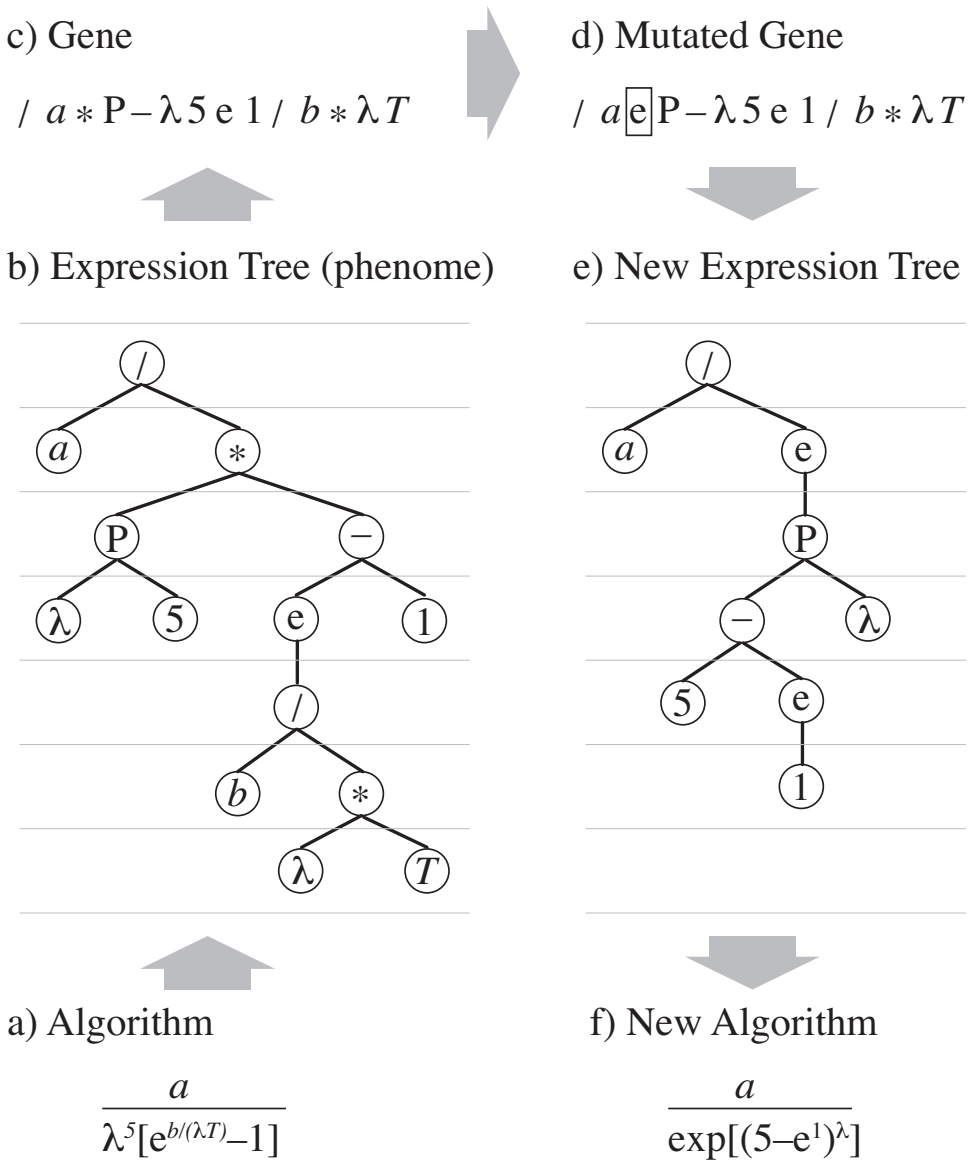


Figure 4.1: Illustration of GEP coding. (a) Planck's law as a mathematical formula. The variables are wavelength λ and temperature T, and the Planck constants are a and b. (b) Expression-tree representation (the phenome: physical representation). The basic operators are $(*, /, -)$, and the basic functions are power function $P(x,y) = x^y$, and e representing $\exp(x)$. (c) GEP coding (the gene: info that describes the phenome). The code is created by reading the expression tree as a book (i.e., NOT following the node connections, but reading each line from left to right, from the top line to the bottom). (d) Mutation of the 3^{rd} character in the gene. (e) New expression tree built from the mutated gene. This is possible because the arity of each basic function is known; e.g., P takes two arguments, but e takes only one. (f) The corresponding new math formula.

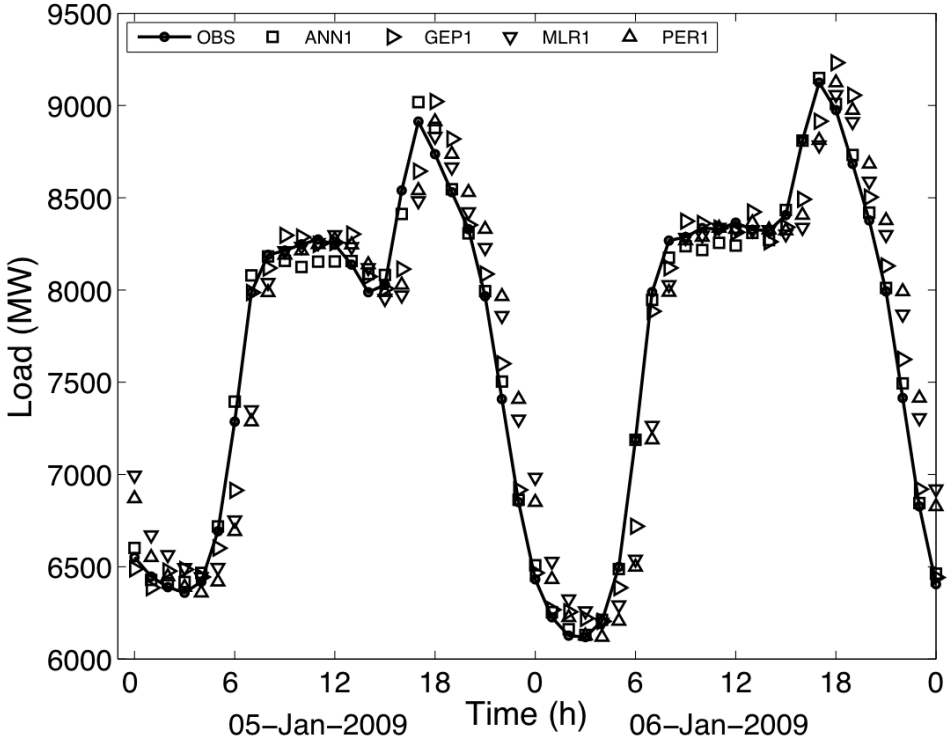


Figure 4.2: A two-day sample showing observed (OBS) vs. forecast loads using only the first stage of regressions. These forecasts are based on stage-one regressions that were trained on the full multiyear training data set.

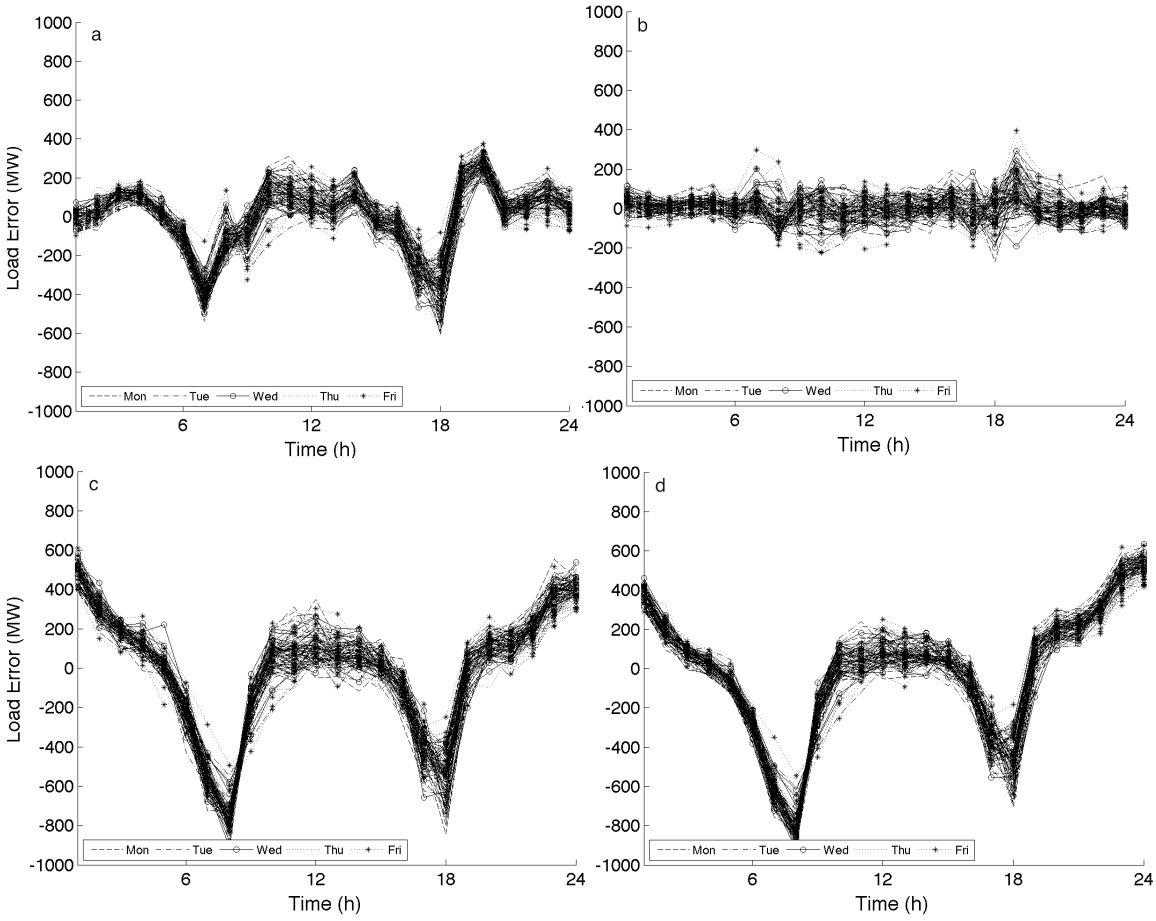


Figure 4.3: Load-forecast error vs. hour of the day, for every weekday in every January of the training data set, after stage one. (a) GEP_1. (b) ANN_1. (c)MLR_1. (d) PER_1.

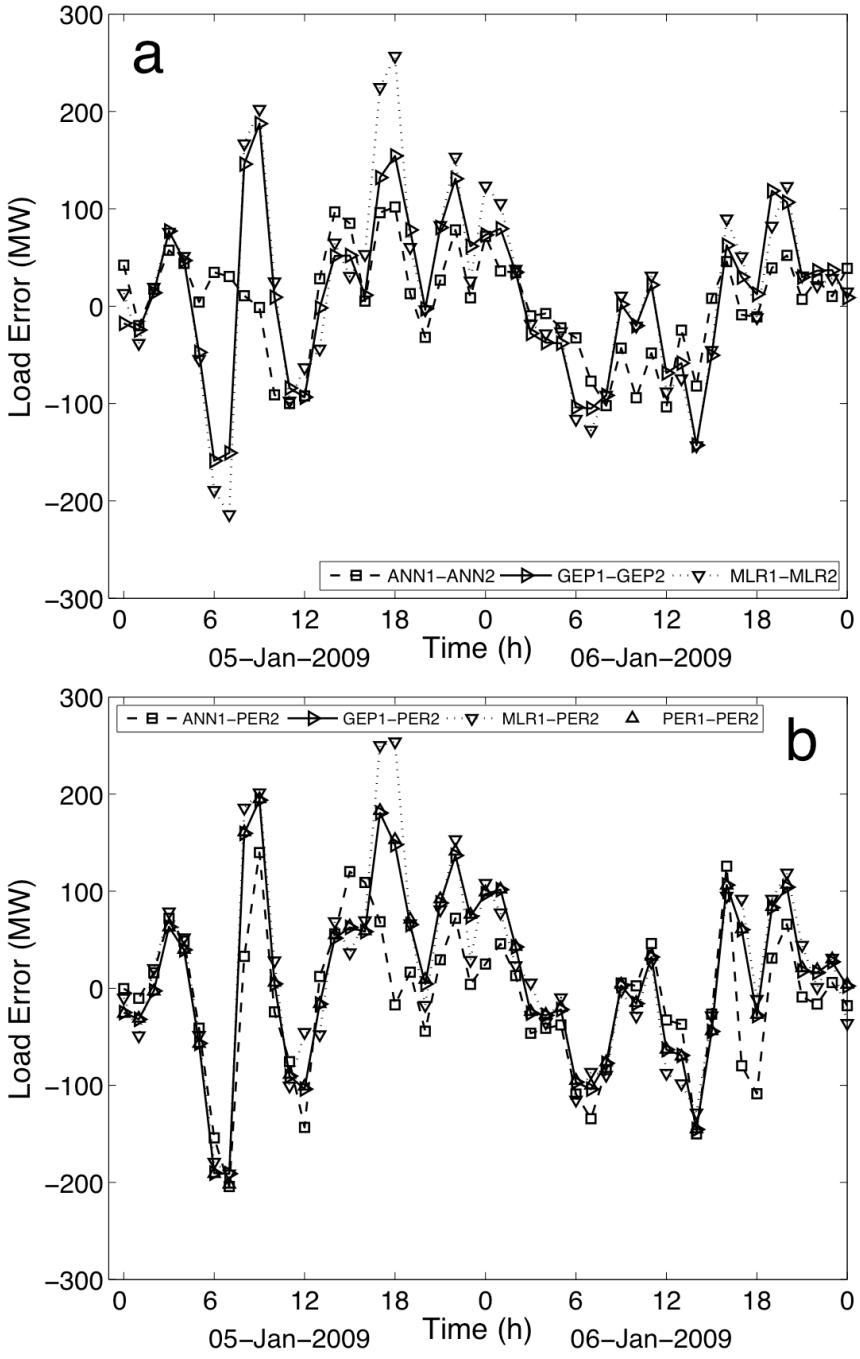


Figure 4.4: Two-day sample of errors after both regression stages. (a) ANN_1-ANN_2, GEP_1-GEP_2, and MLR_1-MLR_2. (b) ANN_1-PER_2, GEP_1-PER_2, MLR_1-PER_2, and PER_1-PER_2.

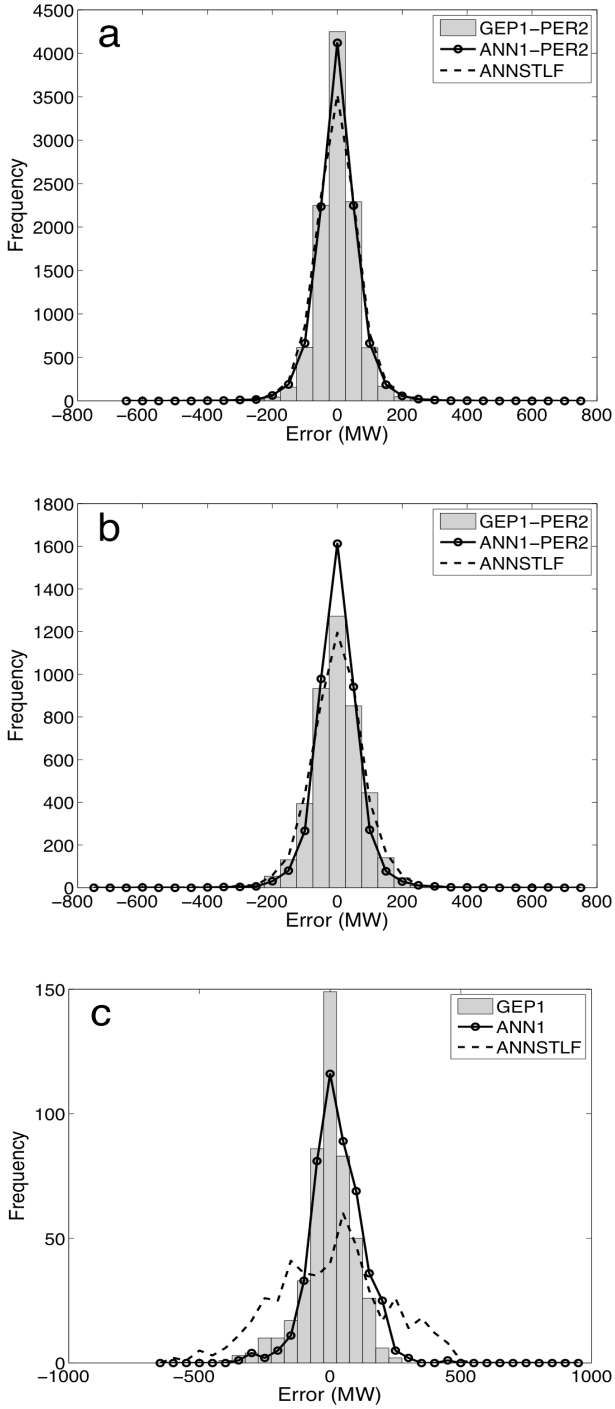


Figure 4.5: ANN_1-PER_2 and GEP_1-PER_2 error distributions for the independent data set. Frequency is the count of hours having that load error. (a)Weekdays. (b) Weekends. (c) Holidays (only first stage). Also shown are the errors from the operational ANNSTLF forecasts.

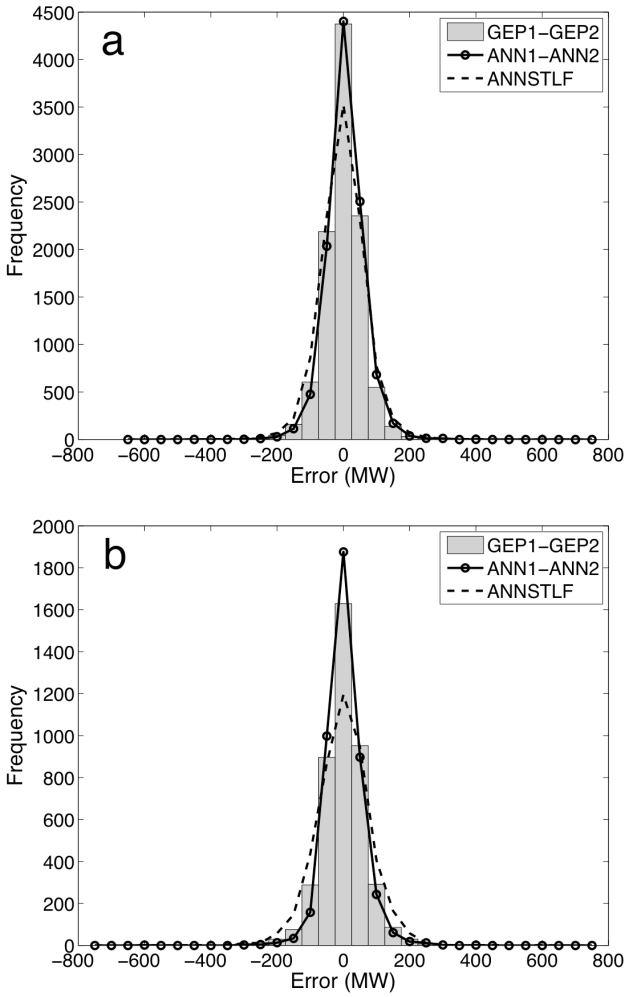


Figure 4.6: ANN_1-ANN_2 and GEP_1-GEP_2 error distributions for the independent data set. (a) Weekdays. (b) Weekends. Also shown are the errors from the operational ANNSTLF forecasts.

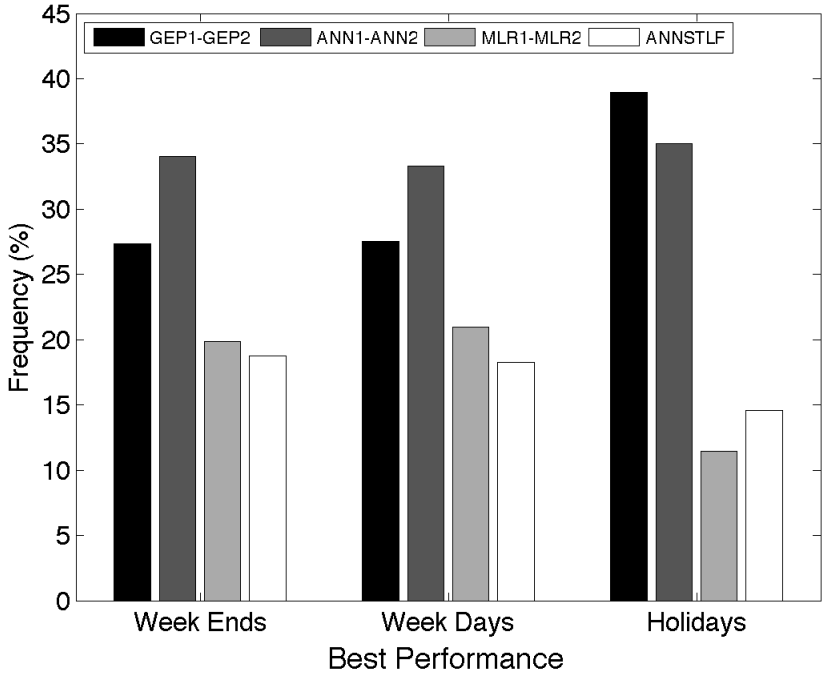


Figure 4.7: Proportion of time that any method was better than the others.

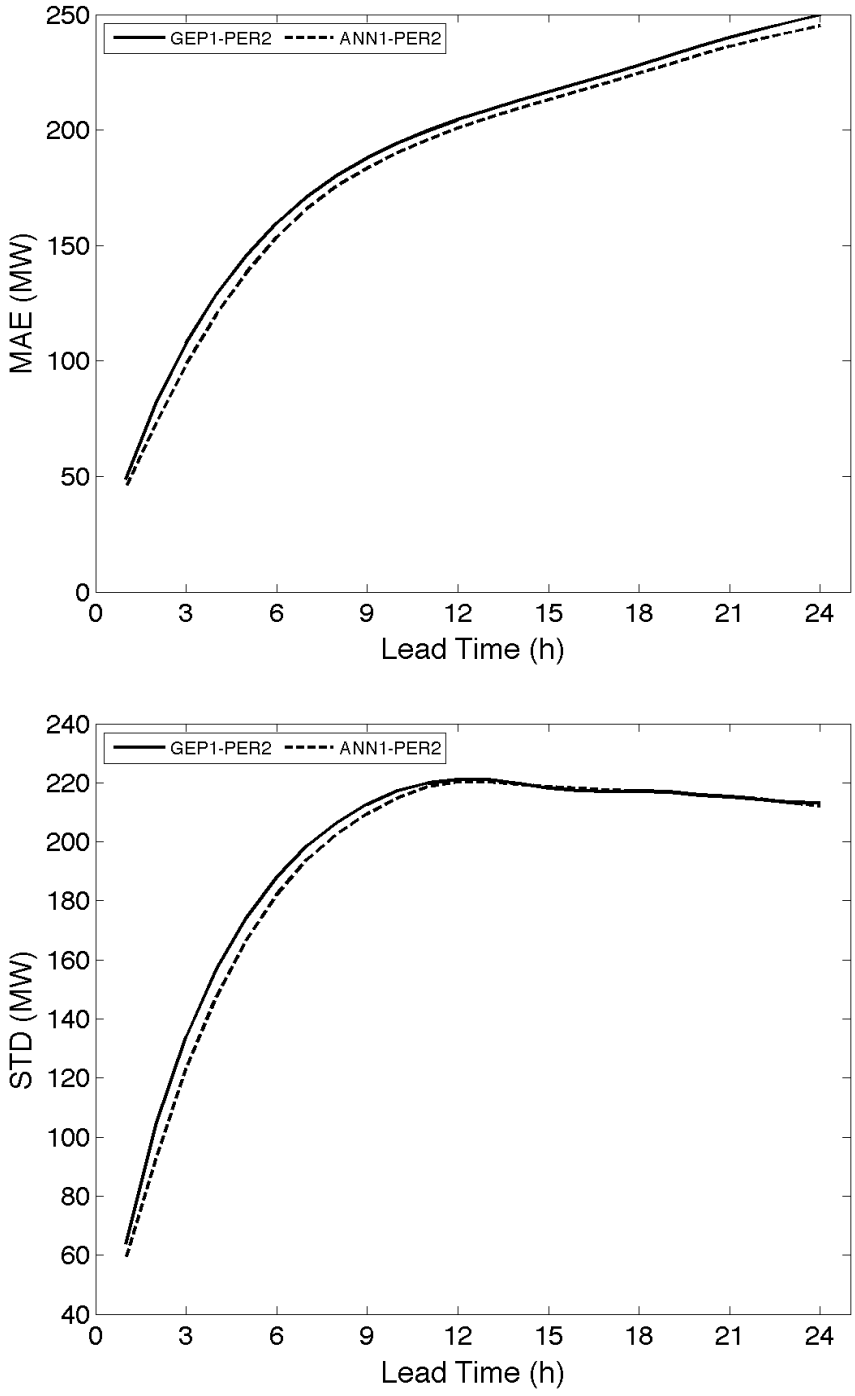


Figure 4.8: Increase of forecast error with lead time, for forecasts made by iteratively re-using the 1-h-ahead load forecasts with the observed temperatures of each hour. MAE is mean absolute error, and STD is standard deviation.

1 APRIL 2009 TO 31 DECEMBER 2010 1 HOUR LOAD FORECASTS		ME (MW)	MAE (MW)	STD (MW)	r^2
GEP_1-GEP_2	WEEK DAYS	-0.32	42	57	0.997
	WEEK ENDS	0.11	48	67	0.995
	HOLIDAYS	-5.0	68	96	0.991
GEP_1-PER_2	WEEK DAYS	0.02	44	61	0.997
	WEEK ENDS	0.11	53	73	0.995
	HOLIDAYS	-5.0	68	96	0.991
ANN_1-ANN_2	WEEK DAYS	5.67	41	56	0.997
	WEEK ENDS	2.33	40	55	0.997
	HOLIDAYS	25.8	76	98	0.991
ANN_1-PER_2	WEEK DAYS	0.01	45	63	0.997
	WEEK ENDS	-0.08	48	67	0.995
	HOLIDAYS	25.8	76	98	0.991
PER_1-PER_2	WEEK DAYS	-0.03	43	60	0.998
	WEEK ENDS	-0.10	56	74	0.994
	HOLIDAYS	4.71	174	219	0.953
MLR_1-PER_2	WEEK DAYS	-0.07	54	73	0.995
	WEEK ENDS	-0.18	65	86	0.992
	HOLIDAYS	3.73	170	212	0.955
MLR_1-MLR_2	WEEK DAYS	-1.36	51	68	0.996
	WEEK ENDS	0.06	61	81	0.993
	HOLIDAYS	3.74	170	212	0.955
ANNSTLF	WEEK DAYS	-2.66	83	114	0.989
	WEEK ENDS	-6.23	81	114	0.986
	HOLIDAYS	-4.48	164	218	0.956

Table 4.1: Verification statistics on independent data after both stages

1 APRIL 2009 TO 31 DECEMBER 2010 1 HOUR LOAD FORECASTS		ME (MW)	MAE (MW)	STD (MW)	r^2
GEP_1- PER_2 SHORT	**WEEK DAYS**	0.00	43	60	0.997
	WEEK ENDS	–0.10	53	72	0.995
GEP_1-PER_2	**WEEK DAYS**	0.02	44	61	0.997
	WEEK ENDS	0.11	53	73	0.995
ANN_1- PER_2 SHORT	**WEEK DAYS**	–0.06	49	67	0.996
	WEEK ENDS	–0.18	52	71	0.995
ANN_1-PER_2	**WEEK DAYS**	0.01	45	63	0.997
	WEEK ENDS	–0.08	48	67	0.995

Table 4.2: Electric load forecast errors for the verification (scoring) data set after the full two-stage regression, but where either short or long data sets were used for training.

1 April 2009 to 31 December 2010 1 Hour Load Forecasts		**ME (MW)**	**MAE (MW)**	**STD (MW)**	$\mathbf{r^2}$
ANN_1	**Week Days**	9.2	51.	70.	0.996
	Week Ends	11.	46.	64.	0.996
	Holidays	26.	76.	98.	0.991
GEP_1	**Week Days**	0.9	81	113	0.989
	Week Ends	3.5	74.	103.	0.989
	Holidays	-5.5	69.	96.	0.991
MLR_1	**Week Days**	1.3	191.	259.	0.941
	Week Ends	6.4	161.	206.	0.954
	Holidays	3.8	170.	212.	0.955
PER_1	**Week Days**	-2.1	197.	274.	0.935
	Week Ends	4.2	166.	214.	0.951
	Holidays	4.9	173.	219.	0.953
ANN_1 Short	**Week Days**	12	63	84	0.994
	Week Ends	7.0	49	67	0.995
GEP_1 Short	**Week Days**	0.2	78.	112	0.989
	Week Ends	-0.3	73.	101.	0.989

Table 4.3: Verification statistics on independent data for the first stage

Chapter 5

Saturated Pseudoadiabats −A Non-iterative Approximation

Two non-iterative approximations are presented for saturated pseudoadiabats (also known as moist adiabats). One approximation determines which moist adiabat passes through a point of known pressure and temperature, such as through the lifting condensation level on a skew-T or tephigram. The other approximation determines the air temperature at any pressure along a known moist adiabat, such as the final temperature of a rising cloudy air parcel. The method used to create these statistical regressions is a relatively new variant of evolutionary algorithms called gene-expression programming. The correlation coefficient between the resulting non-iterative approximations and the iterated data such as plotted on thermodynamic diagrams is over 99.97%. The mean absolute error is 0.28°C and the root mean squared error is 0.44°C within a thermodynamic domain bounded by $-30 < \theta_w \leq 40$°C, $P > 20$ kPa, and $-60 \leq T \leq 40$°C, where $[\theta_w,\ P,\ T]$ are [wet-bulb potential temperature, pressure, and air temperature].

5.1 Introduction and overview of pseudoadiabatic theory

5.1.1 Dry adiabats

In a thermodynamic diagram such as the skew-T log-P in Figure 5.1, the "dry" adiabat is described by a non-iterative equation (Emanuel 1994, eq. 4.2.11):

$$T_f = T_i(P_f/P_i)^{b(R_d/C_{pd})} \tag{5.1}$$

By non-iterative, we mean that the absolute temperature T_f at any final pressure P_f can be solved directly knowing the initial pressure and absolute temperature (P_i, T_i). R_d is the gas constant for dry air, C_{pd} is the specific heat at constant pressure for dry air, and their dimensionless ratio is $R_d/C_{pd} = 0.28571$. The word "dry" means unsaturated humid air. For the special case of $P_f = 100$ kPa, then T_f is defined as the potential temperature θ, and this theta can be used as a label for the

dry adiabat.

Factor b in Eq. 5.1 explains the variations of C_p and R with humidity. Emanuel (1994, eq. 4.2.11) gives dimensionless factor b as

$$b = (1 + r/\varepsilon)/(1 + r/c) \approx (1 - 0.24r) \tag{5.2}$$

where r is the water-vapor mixing ratio in unit $g_{water}g^{-1}_{dry\ air}$, $\varepsilon = R_d/R_v = 0.622\ g_{water}g^{-1}_{dry\ air}$ is the ratio of ideal gas constants for dry air and water vapor, and $c = C_{pd}/C_{pv} \approx 0.5427\ g_{water}g^{-1}_{dry\ air}$ is the ratio of specific heats at constant pressure for dry air and water vapor at 0°C. In the atmosphere r is usually less than $0.04\ g_{water}g^{-1}_{dry\ air}$, hence $b \approx 1$ with an error of less than 1% (Bolton, 1980). From Eq. 5.1, the slope of the dry adiabat in pressure coordinates is

$$\partial T/\partial P = bR_dT/(C_{pd}P) \tag{5.3}$$

In height coordinates, this vertical temperature gradient (Emanuel 1994, eq. 4.7.4) is

$$\partial T/\partial z = -(g/C_{pd})b(T/T_v) \equiv -\Gamma_d \tag{5.4}$$

where $g = 9.8\ \mathrm{ms}^{-1}$ is gravitational acceleration magnitude near the earth's surface, and the virtual absolute temperature is $T_v = T[(1 + r/\varepsilon)/(1 + r)] \approx T(1 + 0.608r)$. To good approximation, the dry adiabatic lapse rate is $\Gamma_d \approx (g/C_{pd}) \approx 9.76\ ^\circ\mathrm{C/km}$.

5.1.2 Saturated pseudoadiabats

Unfortunately, the saturated pseudoadiabat (also sometimes called the wet adiabat or moist adiabat) does not have such a direct definition. Instead, the slope $\partial T/\partial z$ or $\partial T/\partial P$ of the saturated pseudoadiabat can be found from conservation of moist entropy as a function of temperature and saturated mixing ratio r_s, which itself is a nonlinear function of T and P along that adiabat. The ideal gas constant R and the specific heat of air at constant pressure C_p are also functions of r_s. Hence, the equation for the pseudoadiabat must be iterated in small steps of ΔP from an initial (P_i, T_i) to find the temperature T_f at final pressure P_f. Such iteration can be computationally expensive.

For a pseudoadiabatic process, where all liquid water is assumed to fall out as soon as it forms, Emanuel (1994, eq. 4.7.3 and 4.7.5) gives the saturated lapse rate ($\Gamma_s = -\partial T/\partial z$) as

$$\frac{\Gamma_s}{\Gamma_d} = \frac{1 + \frac{L_v r_s}{R_d T}}{1 + \frac{L_v^2 r_s \varepsilon b}{R_d C_{pd} T^2}} \tag{5.5}$$

where Γ_d is from Eq. 5.4, and the saturation mixing ratio is

$$r_s = \varepsilon e_s/(P - e_s). \tag{5.6}$$

The saturation vapor pressure e_s can be found from the Clausius-Clapeyron equation, from Tetens' equation, or from Bolton's equation (Bolton 1980, eq. 10). Emanuel (1994) uses Bolton's equation:

$$e_s = e_0 \exp\Big[\frac{17.67(T - 273.15K)}{T - 29.65K}\Big] \tag{5.7}$$

for T in Kelvin, and where the reference vapor pressure is $e_0 = 0.6112$ kPa at 0°C. The other parameters in Eq. 5.5 as given by Bolton (1980) are the latent heat of vaporization $L_v (\approx 2.501 \times 10^6$ Jkg^{-1} at 0°C), the gas constant for dry air $R_d = 287.053$ JK^{-1}kg^{-1}, and the specific heat at constant pressure for dry air C_{pd}(1004 JK^{-1}kg^{-1} at 0°C).

The hydrostatic equation and the ideal gas law can be used to rewrite Eq. 5.5 in terms of pressure:

$$\frac{\partial T}{\partial P} = (\frac{b}{P})\frac{R_d T + L_v r_s}{C_{pd} + \frac{L_v^2 r_s \varepsilon b}{R_d T^2}} \tag{5.8}$$

Because pressure decreases as height increases, Eq. 5.8 gives a vertical temperature gradient of the proper sign. Eq. 5.5 and Eq. 5.8 are similar to ones presented by Bohren and Albrecht (1998, eq. 6.111) and Stull (2011b, eqs. 4.37 and 4.38). Further simplifications occur by using $b \approx 1$.

Thus, the iterative solution proceeds as follows. For any starting $[P_1, T(P_1)]$ such as at an initial point $(P_i, T_i) = (100$ kPa, $\theta_w)$ on the saturated adiabat, solve Eq. 5.7 for e_s, then solve Eq. 5.6 for r_s, then solve Eq. 5.2 for b (using r_s instead of r) or use the approximation $b \approx 1$, then solve Eq. 5.8 for $\partial T/\partial P$. Next, assume that the saturated adiabat is approximately linear over a very small increment of pressure $\Delta P = P_2 - P_1$, giving $T(P_2) \approx T(P_1) + (P_2 - P_1)\partial T/\partial P$. This new temperature and pressure can be used as input to the next small increment, with the process repeated until the final destination pressure P_f is reached.

If Eq. 5.8 is initialized at a non-reference altitude (i.e., for $P \neq 100$ kPa) and iterated to reach a final pressure of $P_f = 100$ kPa, then the final temperature T_f is defined as the web-bulb potential temperature θ_w, which can be used as a label for that saturated pseudoadiabat (Figure 5.2). If Eq. 5.8 is iterated to the top of the atmosphere ($P_f = 0$ kPa) then its final potential temperature is defined as the equivalent potential temperature θ_e, which is an alternative label (Figure 5.2). Bolton (1980, eq. 40) and Emanuel (1994, eq. 4.7.10) give the following empirical relationship between θ_e and θ_w:

$$\theta_e = \theta_w \exp\{r_{s0}(1 + c_1 r_{s0})(c_2/\theta_w - c_3)\} \tag{5.9}$$

for $r_{s0}(g_{water\ vapor}g^{-1}_{dry\ air})$ = saturation mixing ratio at $P = 100$ kPa and $T = \theta_w$, $c_1 = 0.81\ g_{dry\ air}g^{-1}_{water\ vapor}$, $c_2 = 3376$ K $g_{dry\ air}g^{-1}_{water\ vapor}$, and $c_3 = 2.54 g_{dry\ air}g^{-1}_{water\ vapor}$. When we did some test iterations of Eq. 5.8 from $P_i = 100$ kPa to $P_f = 10$ kPa with an increment of $\Delta P = 0.05$ kPa, we indeed found the saturated pseudoadiabat to be asymptotic to a dry adiabat having $\theta = \theta_e$ given by Eq. 5.9.

5.1.3 Goal and motivation

The goal of this paper is to present non-iterative approximations to the saturated pseudoadiabats. The motivation is to accelerate numerical weather prediction (NWP), where moist-adiabatic computations must be preformed in each grid column for each time step. As NWP models continue to achieve finer grid resolution and require larger numbers of smaller time steps, the number of required calculations increases exponentially. Such pseudoadiabatic computations provide important input to other NWP subroutines for buoyancy, convection, turbulence, cloud microphysics, and precipitation. Computational savings by replacing repeated iterative computations with direct computation could enable forecasts to finish sooner, to the benefit of commerce and society.

Section 5.2 examines differences between a few published thermodynamic diagrams, and describes which data will be used for this study. Section 5.3 explains the methodology of symbolic regression used to find predictands from predictors. Section 5.4 gives the regression result for the wet-bulb potential temperature θ_w as a function of pressure and temperature (P,T). This identifies which specific pseudoadiabat (θ_w = constant) is to be used for saturated ascent or descent. Section 5.5 gives an expression for T as a function of θ_w and destination pressure P. Conclusions are in section Section 5.6 .

5.2 Data

To create a non-iterative relationship for the pseudoadiabat, we need to regress our functions against training data of $\theta_w(P,\ T)$. Although this data could be found iteratively from the pseudoadiabatic theory of Section 5.1, it could alternately be found from data corresponding to traditional printed thermodynamic diagrams. Either approach could yield useful input data − we chose to fit the traditional thermodynamic diagrams. These diagrams, and their differences from the theory of Section 5.1, are explained next.

5.2.1 Diagram types and their operational implementations

Thermodynamic information including dry and saturated pseudoadiabats can be plotted in a variety of formats: emagram, skew-T log-P, tephigram, Stüve, θ-z, θ_w-r_T, and others (Hess, 1959; Emanuel, 1994; Ambaum, 2010; Stull, 2011a). Identical thermodynamic information can be plotted with

identical accuracy on any of these diagrams.

Historically, different forecast centers chose to use different formats, which they printed on large-size paper for use in operational plotting of atmospheric soundings. For example, the skew-T diagram is commonly used in the USA (DoD/USAF 1978; AWS 1990), the tephigram dominates in many British commonwealth countries (Ambaum, 2011) including Canada (EC, 1976), the Stüve has been used in some Germanic countries and by some aviation authorities (ATAA, 1942). These three printed diagrams are pseudoadiabatic diagrams, where all liquid water is assumed to fall out immediately. Printed soundings are rarely used any more except in education. Instead computer-generated soundings are freely available via the internet in a wide variety of formats. Nonetheless, many meteorologists picture the high-resolution printed diagrams in their minds as a reference when they view and interpret the coarser-resolution computer-generated soundings.

However, there is slight disagreement between the printed diagrams listed above. Most agree with each other in the bottom half of the troposphere, but there is some divergence in values higher in the atmosphere for warm saturated pseudoadiabats. For example, following the $\theta_w = 30°$C saturated pseudoadiabat up to a pressure of P = 20 kPa yields temperatures of (−33.3, −32.6, −33.2°C) on the (USAF skew-T, EC tephigram, ATAA Stüve). Some of these differences might be related to drawing errors of the original master diagram, and some are likely due to small differences in values of thermodynamic parameters. Bohren and Albrecht (1998) demonstrate that the $\theta_w = 25°$C saturated adiabat can be 3°C warmer at P = 20 kPa than the corresponding pseudoadiabat, and the ice-saturated pseudoadiabat can be 2°C warmer than the water-saturated pseudoadiabat.

Given typical rawinsonde uncertainties on the order of 0.5°C near the top of the troposphere (Vaisala, 2010) and larger errors in their spatial representativeness, the small differences between different diagrams and their associated assumptions are relatively negligible.

5.2.2 Data selection and preparation

For this study, we need to use a data field (a 2-D array) of $\theta_w(P,T)$ values as input for our regressions. We generate this input data iteratively. Instead of using Eq. 5.6 to Eq. 5.8, we use the following slightly modified pseudoadiabatic equations and constants that we empirically found to provide a better fit to the printed skew-T, tephigram and Stüve diagrams:

$$\partial T/\partial P = \left(\frac{1}{P}\right)\frac{R_d T + L_v r_s}{(C_{pd} - C_w r_s) + \frac{L_v^2 r_s \varepsilon}{R_d T^2}} \tag{5.10}$$

with $T_0 = 273.16$ K, $L_v = 2.50 \times 10^6$ Jkg^{-1}, $R_d = 287.04$ JK^{-1}kg^{-1}, $\varepsilon = 0.62197 g_{water} g_{dry\ air}^{-1}$, $C_{pd} = 1005$ JK^{-1}kg^{-1}, $C_w = 4218$ JK^{-1}kg^{-1} (the specific heat for liquid water), and where Eq. 5.6

is used for r_s. For the saturation vapor pressure in Eq. 5.6 we use Tetens' formula (Bolton 1980, eq. 8; Stull 2011b):

$$e_s = e_0 \exp\left[17.26939(T - T_0)/(T - 35.86)\right] \tag{5.11}$$

which was modified to use T in Kelvin, and which uses $e_0 = 0.61078$ kPa.

For comparison, when Eq. 5.8 and Eq. 5.10 are iterated in 0.2 kPa increments along the $\theta_w =$ 30°C saturated adiabat from initial point (P_i, T_i) = (100 kPa, 30°C) to final point (P_f, T_f) = (20 kPa, T_f), the resulting final temperatures are [(Eq. 5.8), (Eq. 5.8 with b = 1), (Eq. 5.10)] = $[-31.36, -31.46, -33.07°\text{C}]$. This last value (from Eq. 5.10) is within the range of $-33.2 \leq T_f \leq -32.6°\text{C}$ from the printed diagrams. Similarly, Eq. 5.10 was found to fit the printed diagrams for other saturated pseudoadiabats and other ending pressures.

Eq. 5.10 is asymptotic to a different dry adiabat than is Eq. 5.8. We empirically found the following relationship between θ_e and θ_w for the thermodynamic diagrams studied here:

$$\theta_e = \theta_w \exp\left[r_{s0}(1 + c_1 r_{s0})(c_2/\theta_w)\right] \tag{5.12}$$

for r_{s0} ($g_{water\ vapor} g^{-1}_{dry\ air}$) = saturation mixing ratio at P = 100 kPa and $T = \theta_w$, $c_1 = 0.81\ g_{dry\ air} g^{-1}_{water\ vapor}$, $c_2 = 2555$ K $g^{-1}_{water\ vapor}$, and where we used Tetens' formula (Eq. 5.11) for saturation vapor pressure. Although we do not use Eq. 5.12 for our non-iterative approximations, we present it here because some readers may prefer to identify the saturated pseudoadiabat by θ_e rather than θ_w.

5.3 Methodology

5.3.1 Two algorithms needed

When using a thermo dynamic diagram or its computational equivalent, two operations are often needed with regard to saturated-adiabatic processes (i.e., for a cloudy air parcel). First, one needs to determine which saturated pseudoadiabat the air parcel follows from the initial state (P_i, T_i). Second, after following this pseudoadiabat to some final pressure P_f, one needs to know the final temperature T_f of the air parcel. Needed for the first operation is $\theta_w(P, T)$, while needed for the second is $T(P, \theta_w)$. These two operations are analogous to entering and exiting a highway, where the highway is the saturated pseudoadiabat (Figure 5.3).

Users of thermo dynamic diagrams often determine the initial conditions for the first operation from the lifting condensation level (LCL) of a hypothetical rising unsaturated air parcel $(P_i, T_i) = (P_{LCL}, T_{LCL})$. Knowing the initial conditions $(P*, T*, T_d*)$ of an unsaturated air parcel where T_d is

the dew-point temperature, then the LCL conditions can be estimated from:

$$P_{LCL} \approx P* \left[1 - a(\frac{T* - T_d*}{T*})\right]^{C_{pd}/R_d} \tag{5.13}$$

where a = 1.225 is a dimensionless empirical constant (Stull, 2011a) and $C_{pd}/R_d = 3.50$ (dimensionless). T_{LCL} can then be solved from Eq. 5.1 by using dry-adiabatic initial state $(P*, T*)$ and setting the final state to (P_{LCL}, T_{LCL}). Eq. 5.13 is not part of the current study − indeed other researchers might use more precise methods to find the LCL or other initial conditions to define θ_w. We will assume that the user provides appropriate initial conditions (P_i, T_i).

Each operation $[\theta_w(P,T)$ or $T(P,\theta_w)]$ requires a separate regression, but both regressions are fit to the same data set that is created iteratively with Eq. 5.10. If one of the regressions had resulted in a simple algorithm, then it could have been solved analytically (i.e., inverted) for the other operation without requiring a second regression. However, neither algorithm is simple, so both regressions are required.

We use an evolutionary regression method where not only the constants (or weights) in the algorithm are found, but also the functional form of the algorithm can vary (namely, it is not limited to any a-priori algorithm form such as a polynomial or a neural network). This regression is performed using gene-expression programming.

5.3.2 Gene expression programming

Gene-expression programming (GEP) is an evolutionary method that can be used to find a best-fit function to data. Recently devised by Ferreira (2006), GEP is an efficient variant of genetic programming and genetic algorithms, where candidate functions evolve via various forms of mutation, and compete via a computational natural selection until the fittest candidate (with the lowest verification error) is found. GEP can explore a wide range of the function space to find a best fit, and can yield a nonlinear result that would not necessarily have been obvious if the function fit had been attempted manually.

The complexity of an algorithm that can be produced by GEP depends on the specification of which and how many basic functions (e.g., exp, sin, arctan) and operators $(+, -, *, /)$ from which GEP is allowed to draw, and on the specification of the number of components (operators, functions, input predictor variables, and numerical constants) that are allowed to make up the algorithm. In the terminology of GEP, these components are represented by characters in a character-string called a chromosome, where the chromosome consists of substrings called genes that are added together. The user designs the component complexity of GEP based on the complexity of the problem to be solved, and on the desire to create the simplest algorithm that adequately fits the target data. Such

design usually requires some trial and error, which can build on the experience and recommendations of Ferreira (2006). The GEP specifications that we used for saturated pseudoadiabats are given in Section 5.4 and Section 5.5.

The focus of this paper is on the non-iterative pseudoadiabat algorithm that results from using GEP, and not on the inner workings of GEP itself. For details on GEP, the reader is referred to Bakhshaii and Stull (2009, 2012) and Ferreira (2006). We used a commercial computer package called GeneXproTools (Ferreira, 2008) that can run on personal computers. Some recent meteorological applications of GEP include estimating ensemble-average precipitation in mountainous terrain from numerical forecasts (Bakhshaii and Stull, 2009), forecasting minimum daily temperature as a model output statistic (Roebber, 2010), approximating wet-bulb temperature from relative humidity and air temperature (Stull, 2011b), and estimating electric load for a population of 4.5M people (Bakhshaii and Stull, 2012).

The reader is cautioned that the best-fit results that we found using GEP are just two realizations out of perhaps an infinite set of alternatives that could provide nearly identical goodness-of-fit. Such is the quasi-random nature of evolutionary programming. Namely, a different researcher could start with the same training data and the same GEP specifications, yet reach a different and equally good result.

5.4 Wet-bulb potential temperature from pressure and temperature: $\theta_w(P,T)$

5.4.1 GEP set-up

GEP was able to find an algorithm to fit θ_w as a function of initial P and T. We programmed GEP to use a mutating population of 30 individuals that competed over thousands of generations for the best fitness. Each individual had a chromosome made of the sum of 6 genes (this is an experimental chosen number based on try and error), where each gene had a maximum size of 29 components. For this evolution, GEP was allowed to randomly select from the following set of operators and functions during initialization and for subsequent mutations: $+, -, *, /$, sqrt, exp, ln, square, cube, sin, cos, cube root, and arctan.

The computational selection of which individuals survive (some with mutation) into the next generation is based on the fitness f of each individual. This fitness is defined as $f = 1000(1 + \text{RRSE})^{-1}$, where the root-relative square error is

$$\text{RRSE} = \left[\frac{\sum_{i=1}^{n}(E_i - O_i)^2}{\sum_{i=1}^{n}(\bar{O} - O_i)^2}\right]^{1/2} \qquad 5.14$$

In this expression, E_i is the value (θ_w in our case) estimated by the individual algorithm for the i^{th} data point, O_i is the corresponding target value, n is the total number of data points, and the overbar indicates a mean value. *RRSE* can be interpreted as the error variance relative to the variance of the target data around the mean. A perfectly fit individual has a fitness of 1000, while an unfit individual has $f = 0$.

The total target data set consisted of N = 4,397 different values of $\theta_w(P,T)$ as calculated iteratively using Eq. 5.10. These N points spanned the subdomain $T > -60°$C, $-25 \leq \theta_w \leq 40°$C, and $P < 10$ kPa, sketched as the unshaded region in Figure 5.4 The order of these N targets was then randomized, and 10% of the points were selected as the target values O for training GEP (i.e., $n = 0.1\ N$ data points were used during the evolution). The viability of final candidate algorithms that were produced by GEP were tested against the full data set ($n = N$). Such a procedure of testing against a larger data set than is used for training is a method that helps to reduce overfitting (to avoid fitting the noise instead of the signal).

5.4.2 Results

The final result from GEP is the following non-iterative algorithm for determining $\theta_w(P,T)$:

$$\theta_w = g_1 + g_2 + g_3 + g_4 + g_5 + g_6 \tag{5.15a}$$

where the six genes are defined as follows:

$$g_1 = \arctan\{-0.0141748[P^{1/2}(8.114196 + T) + 65.8402]\} \tag{5.15b}$$

$$g_2 = [69.2840 + P^{1/2}]^{1/2} + [6.558563 + (8.3237/P)]^2 \tag{5.15c}$$

$$g_3 = \exp(17.850425/P)\sin[0.0510(T - P)] \tag{5.15d}$$

$$g_4 = 0.00740425(T - 23.9263)P \tag{5.15e}$$

$$g_5 = -0.355695[0.5997 + P - T + \arctan(T)] \tag{5.15f}$$

$$g_6 = 0.357635[0.0922 + \arctan(T)]\sin[(3.877869 + P)^{1/2}] \tag{5.15g}$$

In these equations, θ_w and T must have units of °C, P must have units of kPa, and the trig functions treat their arguments as if they are in radians.

Figure 5.5 shows a comparison of the regressed (non-iterative) approximation for θ_w calculated using Eq. 5.15 vs. the target from the iterative solution to Eq. 5.10. Figure 5.6 shows the corresponding θ_w errors as a function of P and T. These errors are on the same order of magnitude as the

errors caused by not knowing whether the real atmospheric process is adiabatic or pseudoadiabatic. The shaded sub-domains in these figures show where errors were greater than 1°C. Extrapolation of Eq. 5.15 beyond the region for which it was trained can give very large errors, and is not recommended.

Verification was done using 44,824 data points distributed uniformly within the skew-T subdomain outlined with the thick black line in Figure 5.4. The verification results are mean absolute error (MAE) = 0.28°C, root mean squared error (RMSE) = 0.44°C, root relative squared error (RRSE) =0.02, and r^2 = 0.9995 between the iterated target points and the non-iterative approximation, where r is the Pearson correlation coefficient.

5.5 Temperature from pressure and wet-bulb potential temperature: $T(P,\ \theta_w)$

5.5.1 GEP set-up

While GEP was able to find a single algorithm to fit T as a function of θ_w and final P, this algorithm had unacceptably large temperature errors (≥ 1°C over many portions of the thermo dynamic diagram). However, by splitting the problem into three subdomains (Cold: $-30 < \theta_w \leq 4$°C; Warm: $4 < \theta_w \leq 21$°C; and Hot: $21 < \theta_w < 45$°C), we were able to find separate regressions for each subdomain with errors mostly less than 0.5°C. Target data points were again calculated from Eq. 5.10 to fill this expanded domain. We started with a small genome and identical settings for all three sections and increased size of the genome and the number of functions gradually. After experimenting to find the smallest chromosomes and function lists that gave an acceptable fit, we arrived at the following GEP specifications.

For the cold θ_w subdomain, the chromosome was the sum of 6 genes, each gene with a maximum of 29 components. The set of operators and functions available for random initialization and subsequent mutations were: $+, -, *, /$, sqrt, exp, ln, square, cube, sin, cos, cube root, arctan, logistic, and the numerical constants $0, 1, e$, and π. There were 2,015 target data points in this subdomain.

For the warm θ_w subdomain, the chromosome was the sum of 7 genes, each gene with a maximum of 29 components. The set of operators and functions available for random initialization and subsequent mutations were: $+, -, *, /$,sqrt, exp, ln, square, cube, sin, cos, cube root, arctan, inverse, log, and fourth root. In this subdomain there were 2,124 target data points.

For the hot θ_w subdomain, the chromosome was the sum of 7 genes, each gene with a maximum of 29 components. The set of operators and functions available for random initialization and subsequent mutations were: $+, -, *, /$, sqrt, exp, ln, square, cube, sin, cos, cube root, arctan, fourth root,

fifth root, min, max, logistic, and the numerical constants 0, 1, e, and π. In this subdomain there were 2,604 target data points.

5.5.2 Results

For the cold subdomain of $-30 < \theta_w \leq 4$°C, the non-iterative algorithm for determining T(°C) from θ_w(°C) and P (kPa) is:

$$T = g_1 + g_2 + g_3 + g_4 + g_5 + g_6 \tag{5.16a}$$

where the six genes are:

$$g_1 = -20.3313 - 0.0253P \tag{5.16b}$$

$$g_2 = \sin[(\theta_w + P)^{1/2} + (\theta_w/P) + P - 2.8565 \tag{5.16c}$$

$$g_3 = \cos\{19.6836 + [1 + \exp(-\theta_w)]^{-1/3} + P/15.0252\} \tag{5.16d}$$

$$g_4 = 4.4653\ \sin(P)^{1/2} - 71.9358 \tag{5.16e}$$

$$g_5 = \left\{\exp[\theta_w - 2.71828\ \cos(P/18.5219)]\right\}^{1/6} \tag{5.16f}$$

$$g_6 = \theta_w - \sin\left\{[P + \theta_w + \arctan(\theta_w) + 6.6165]\right\}^{1/2} \tag{5.16g}$$

For the warm subdomain of of $4 < \theta_w \leq 21$°C, the non-iterative algorithm for determining T(°C) from θ_w(°C) and P (kPa) is:

$$T = g_1 + g_2 + g_3 + g_4 + g_5 + g_6 + g_7 \tag{5.17a}$$

where the six genes are:

$$g_1 = -9.6285 + \cos\left\{\ln[\arctan(\arctan\{\exp[-9.2121\theta_w/P]\})]\right\} \tag{5.17b}$$

$$g_2 = \theta_w - (19.9563/P)\ \arctan(\theta_w) + \theta_w^{1/2}/(5.47162P) \tag{5.17c}$$

$$g_3 = \sin[\ln(8P^3)]\ln(2\ P^{3/2}) \tag{5.17d}$$

$$g_4 = \theta_w + (P\theta_w - P + \theta_w)/(P - 190.2578) \tag{5.17e}$$

$$g_5 = P - (P - 383.0292)/(15.4014P - P^2) \tag{5.17f}$$

$$g_6 = (1/3)\ln(339.0316 - P) + \arctan(\theta_w - P + 95.9839) \tag{5.17g}$$

$$g_7 = -\ln(P)[298.2909 + 16.5109P]/(P - 2.2183) \tag{5.17h}$$

For the warm subdomain of of $21 < \theta_w < 45$°C, the non-iterative algorithm for determining T(°C) from θ_w(°C) and P (kPa) is:

$$T = g_1 + g_2 + g_3 + g_4 + g_5 + g_6 + g_7 \tag{5.18a}$$

where the six genes are:

$$g_1 = 0.3919\theta_w^{7/3}[P(P + 15.8148)]^{-1} \tag{5.18b}$$

$$g_2 = [19.9724 + (797.7921/P)]\ \sin(-19.9724/\theta_w) \tag{5.18c}$$

$$g_3 = \left\{\ln(-3.927765 + \theta_w + P)\ \cos[\ln(\theta_w + P)]\right\}^3 \tag{5.18d}$$

$$g_4 = \{\exp[(\theta_w + (1 + e^{-P})^{-1})^{1/2} - 1.5603]\}^{1/2} \tag{5.18e}$$

$$g_5 = (P + \theta_w)^{1/2}\exp\{\arctan[(P + \theta_w)/7.9081]\} \tag{5.18f}$$

$$g_6 = \{(P/\theta_w^2)\ \min[9.6112, (P - \theta_w)]\} - 13.7300 \tag{5.18g}$$

$$g_7 = \sin\left\{\sin^3[\min(P, 17.3170)] - P^{1/2} + 25.5113/\theta_w\right\} \tag{5.18h}$$

As before, the trig functions in Eq. 5.16-Eq. 5.18 treat their arguments as if they are in radians, and the "min" function returns the minimum of two arguments. Sample calculations with Eq. 5.15-Eq. 5.18 are given in the Appendix K.

Figure 5.7 shows a comparison of the regressed (non-iterative) approximations for T calculated using Eq. 5.16- Eq. 5.18 vs. the target from the iterative solution to Eq. 5.10. Figure 5.8 shows the corresponding T errors as a function of P and θ_w. Verification was done using 44,824 data points distributed uniformly within the skew-T subdomain outlined with the thick black line in Figure 5.4. The verification results were MAE = 0.27°C, RMSE = 0.36°C, RRSE = 0.01, and r^2 = 0.9998. The shaded sub-domains in Figure 5.7 and Figure 5.8 show where error magnitudes exceed 2°C. Extrapolation of Eq. 5.16- Eq. 5.18 into and beyond these shaded regions is not appropriate, and can give very large errors.

Whenever domains are broken into subdomains for a separate solution, there is concern that values could be discontinuous at the subdomain boundaries. Small T discontinuities are indeed observed at the boundaries ($\theta_w = 4$ and 21°C) in Figure 5.7. The corresponding T errors near those boundaries are nonetheless small − on the order of 0.5°C or less, except for a very small region near (P, T) = (100 kPa, 4°C). To help minimize these errors, we used overlapping subdomains (θ_w = −30°C to +5°C; −5°C to 25°C; and 15°C to 45°C) for each GEP regression. For typical thermodynamic calculations following a saturated air parcel moving adiabatically, recall that θ_w is

constant, so the air parcel would not normally cross a subdomain boundary.

5.6 Discussion and conclusions

Two non-iterative approximations were presented for saturated pseudoadiabats (also known as moist adiabats). One approximation (Eq. 5.15) allows you to determine which moist adiabat passes through a point of known pressure and temperature, such as is drawn on a tephigram, skew-T diagram, or other thermodynamic diagram. The other approximation (Eq. 5.16 - Eq. 5.18) allows you to determine the air temperature at any pressure along a known moist adiabat. The correlation coefficient r between the approximations and the thermo dynamic diagram data is over 99.7%. The difference between the non-iterative approximation and the corresponding iterated target values is less than about 0.5°C over a majority of the skew-T subdomain that is outlined with the thick black line in Figure 5.4. The distribution of errors is plotted in Figure 5.6 and Figure 5.8.

These approximations are based on a statistical regression, not on first principles. They apply only over the domains indicated on Figure 5.6 and Figure 5.8, and would give erroneous results if extrapolated outside that domain. The approximations given as Eq. 5.15 - Eq. 5.18 are not the only ones possible. The random nature of evolutionary programming means that a large number of similarly accurate approximations could likely be found.

Although iteration is not needed when using (Eq. 5.15 - Eq. 5.18) to determine the thermodynamic state of a rising saturated air parcel, some meteorological applications require iteration nonetheless. Examples are buoyancy or stability calculations. Iteration is required for these cases because the environment surrounding the air parcel also changes as the parcel rises. Nonetheless, Eq. 5.15 - Eq. 5.18 allow one to take larger vertical steps such as might correspond to significant levels in a sounding, rather than taking the smaller steps that would have been needed to accurately follow a saturated adiabat using iterative equations.

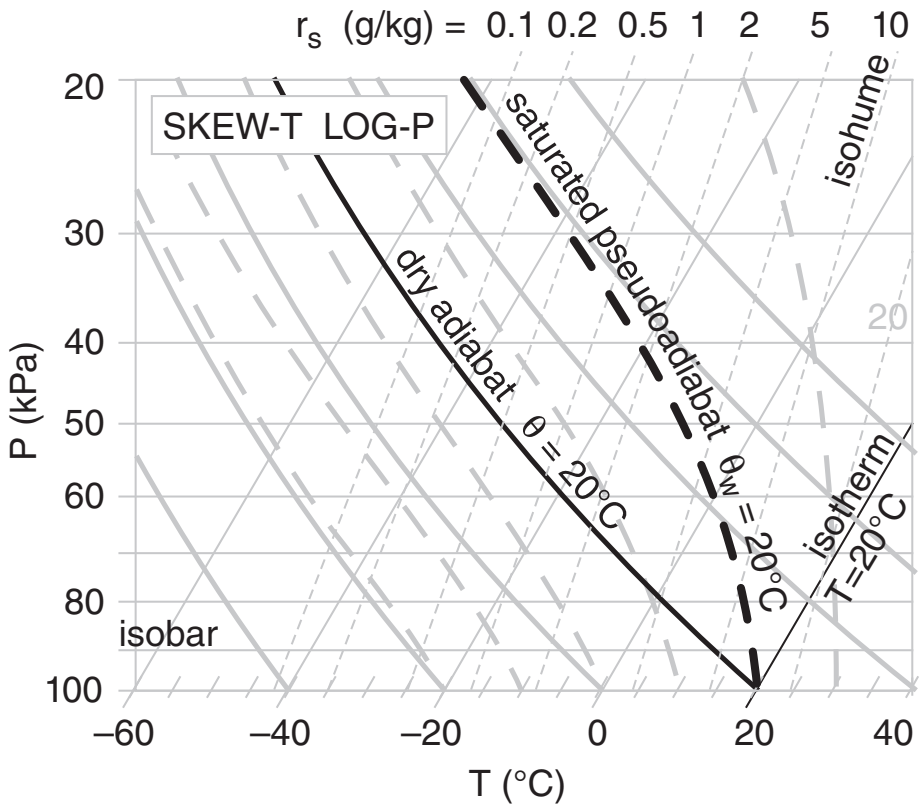


Figure 5.1: Skew-T log-P thermodynamic diagram, with sample isopleth types highlighted in black and labeled. The thick dashed grey lines are a field of saturated pseudoadiabats.

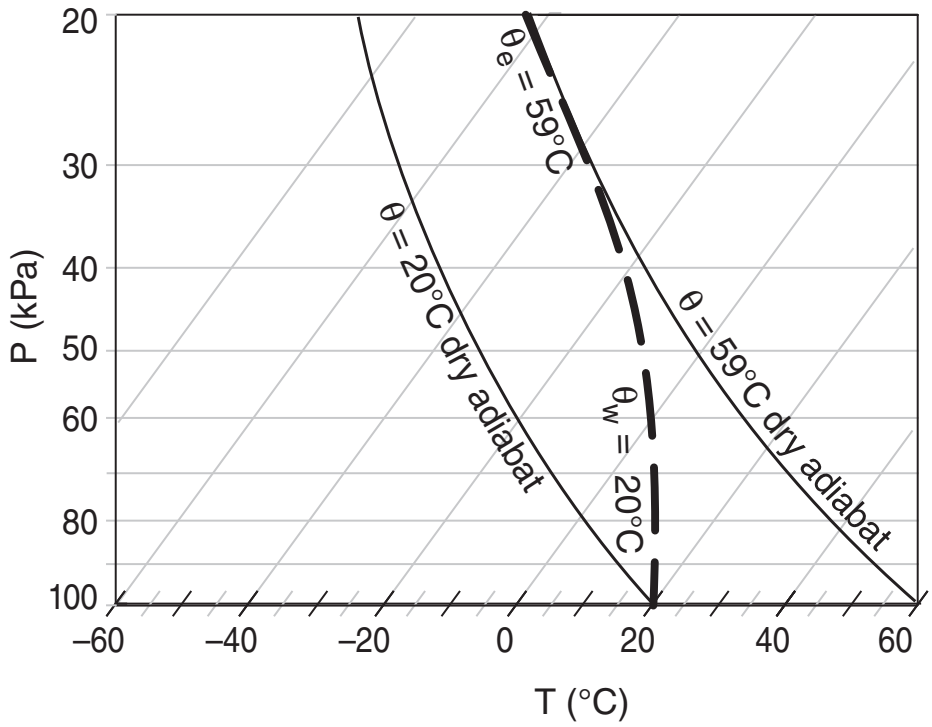


Figure 5.2: Illustration of two methods to label saturated pseudoadiabats (thick dashed line). One is the potential temperature of the pseudoadiabat at a reference pressure of $P = 100$ kPa at the bottom of this diagram, which defines the wet-bulb potential temperature θ_w. The other is the potential temperature towards which the pseudoadiabat asymptotically approaches near the top of this diagram, which defines the equivalent potential temperature θ_e.

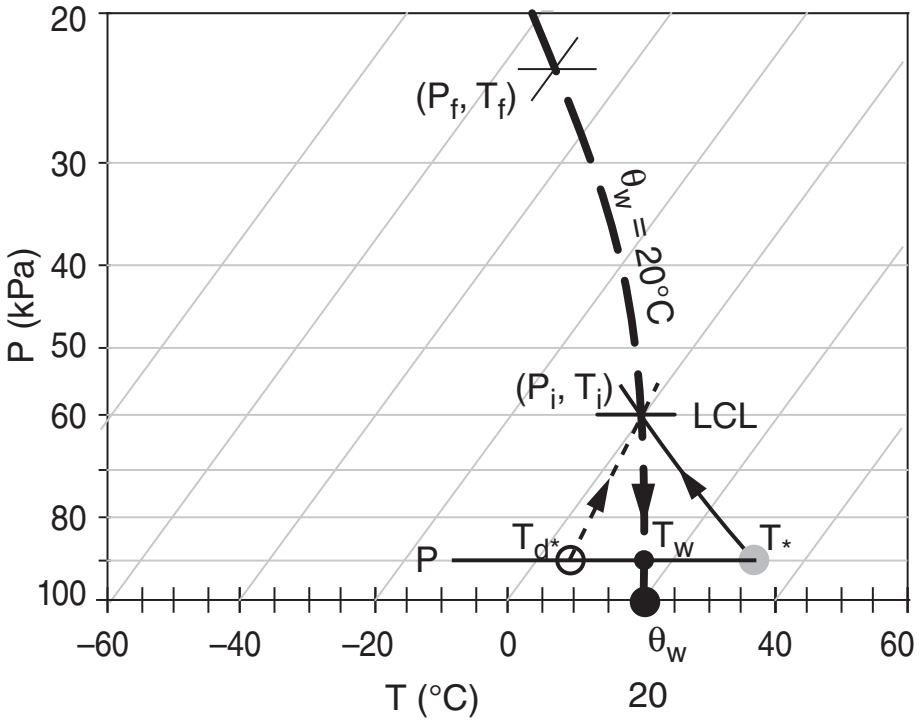


Figure 5.3: Illustration of arbitrary initial (i) and final (f) points along an arbitrary saturated pseudoadiabat (thick dashed line) of wet-bulb potential temperature $\theta_w = 20°\text{C}$. Also shown are the wet-bulb temperature T_w of the initial unsaturated air parcel of temperature T_* and dew point T_d*, and the corresponding lifting condensation level (LCL).

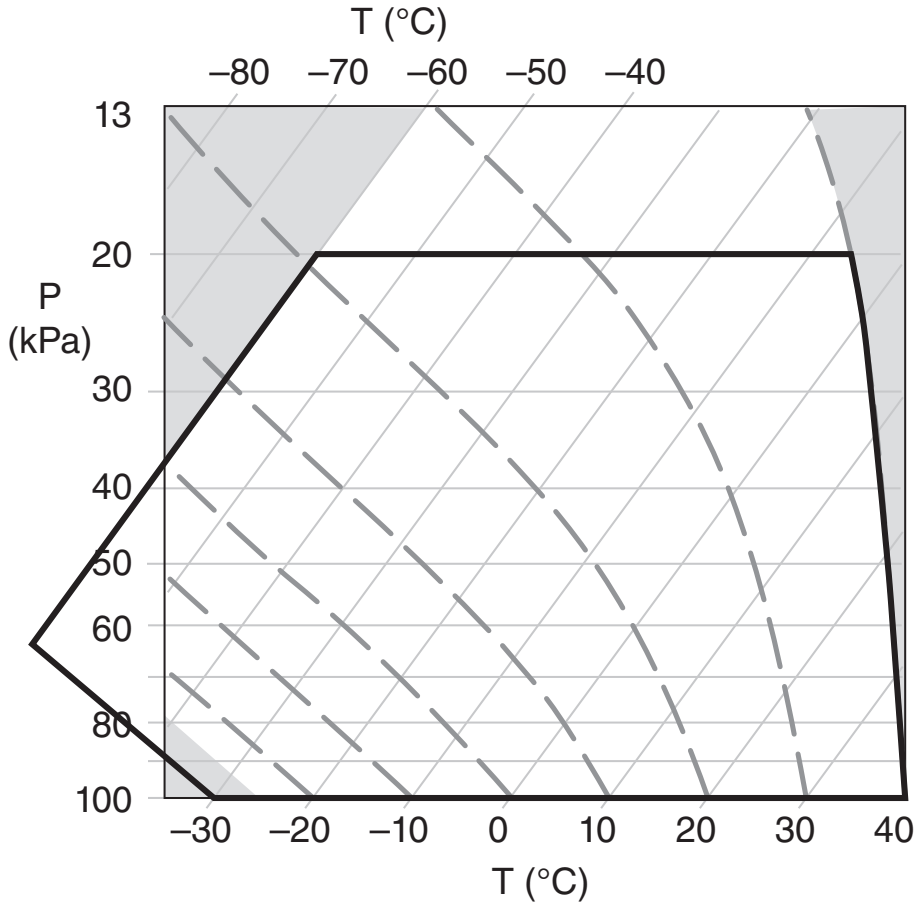


Figure 5.4: Non-shaded region of the skew-T diagram shows most of the subdomain from which data points were taken to train GEP to find $\theta_w(P,T)$. Thick black line outlines the subdomain used to calculate verification scores.

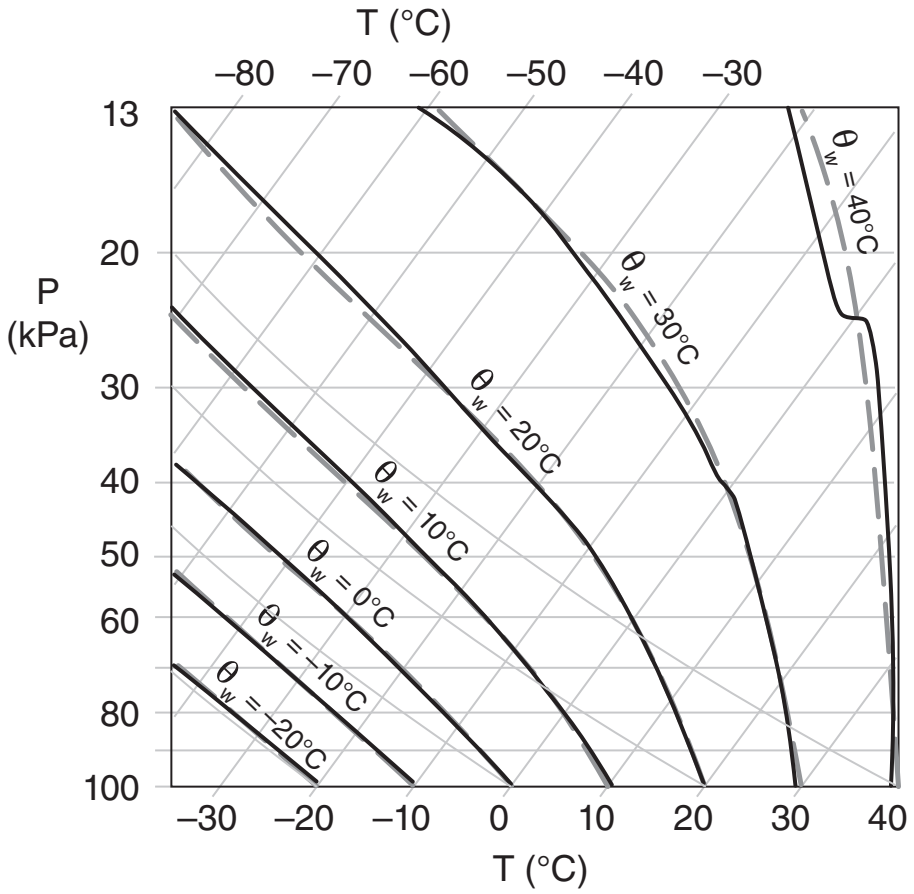


Figure 5.5: Wet-bulb potential temperature θ_w as a function of temperature T and pressure P. The target pseudoadiabats (iterative Eq. 5.10) are plotted as thick grey dashed curves, and the approximations (non-iterative Eq. 5.15) are plotted as thick black curves. Several dry adiabats are plotted for reference as thin grey lines sloping up to the left. The kink in the moist adiabats for hot θ_w values occurs where those adiabats cross the 0 °C isotherm.

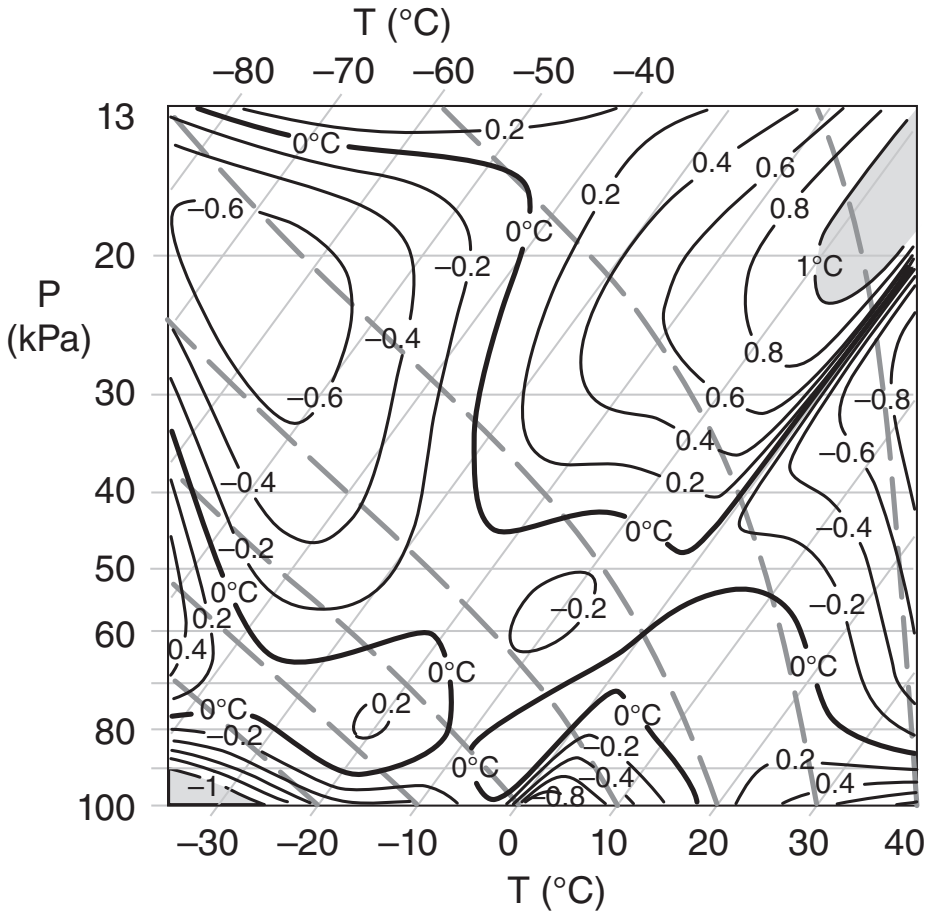


Figure 5.6: Black curves show errors ($\theta_{w\ non-iterative} - \theta_{w\ iterated\ target}$) of wet-bulb potential temperature (°C) as a function of temperature T and pressure P, corresponding to Figure 5.5. The target pseudoadiabats (iterative Eq. 5.10) are plotted as grey dashed curves, labeled at the bottom axis. Error magnitudes greater than 1°C are shaded.

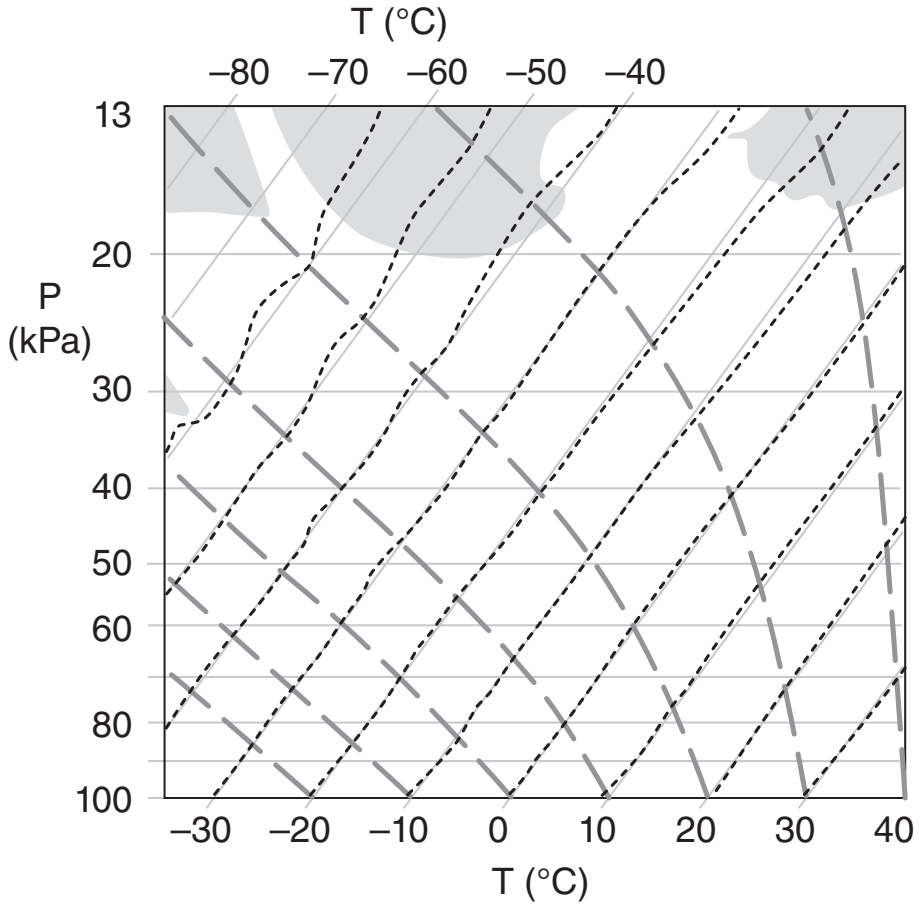


Figure 5.7: Air temperature T as a function of wet-bulb potential temperature θ_w (thick dashed grey curves) and pressure P (thin grey horizontal lines). The target isotherms (iterative Eq. 5.10) are plotted as thin diagonal grey lines, and the approximations (non-iterative Eq. 5.16 - Eq. 5.18) are plotted as dotted black lines. Error magnitudes greater than 2°C are shaded

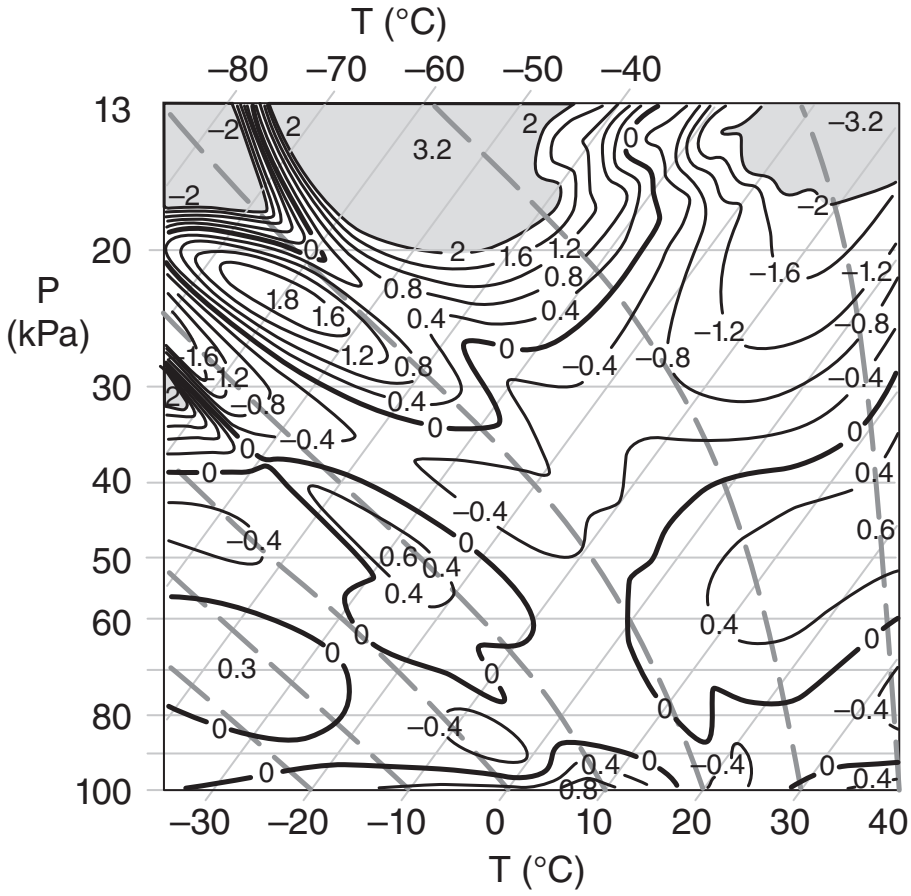


Figure 5.8: Black curves show errors ($T_{non-iterative} - T_{iterated\ target}$) of air temperature (°C) as a function of wet-bulb potential temperature θ_w (thick dashed grey curves) and pressure P (thin grey horizontal lines). These errors correspond to the curves in Figure 5.7. Error magnitudes greater than 2 °C are shaded.

Chapter 6

Summary and Conclusions

Hydrometeorological forecast enhancement using gene-expression programming and its application to clean energy is the goal of this dissertation. The achievements are summarized here.

6.1 Deterministic ensemble forecasts using gene-expression programming

6.1.1 Method

A method is introduced that uses gene-expression programming (GEP) symbolic regression to form a nonlinear combination of ensemble NWP forecasts. The resulting best algorithm yields a deterministic ensemble forecast (DEF) that could serve as an alternative to the traditional ensemble average.

Motivated by the difficulty in forecasting montane precipitation, we test the ability of GEP to produce bias-corrected short-range 24-hour-accumulated precipitation DEFs at 24 weather stations in mountainous southwestern Canada. Input to GEP is 11 limited-area ensemble members from three different NWP models at four horizontal grid spacings. The data consists of 198 quality-controlled observation-and-forecast-date pairs during the two Fall-Winter-Spring rainy seasons of October 2003 through March 2005.

6.1.2 Findings

For our case-study data, we find that GEP DEFs are better than equally-weighted ensemble averages of precipitation for about half of the mountain weather stations tested. A sparse sampling of the multi-model multi-grid-size "ensemble space" spanned by the ensemble members can create a DEF almost as good as the full 11-member ensemble. The best DEF algorithms found using GEP are non-unique and irreproducible, yet can give consistent results that can be used to good advantage at operational forecast centers. In spite of the relative complexity of the DEF algorithms created using GEP, they are non-iterative, and thus computationally efficient to use.

6.1.3 Future work

For some weather stations, the DEF algorithms found by GEP do not give better results than from the pooled (equally weighted) ensemble (PE) average. Although there is no law of computing that says you must use the same bias-corrected ensemble methods at all weather-station locations, more investigation on these sites can be useful. The stations with close results GEP-DEF and PE-DEF might also be used as a tuner for GEP settings.

One can consider the following future research. One can test GEP DEFs over additional data sets, other locations, and for other weather elements. One can include as predictors the raw NWP precipitation forecasts from neighboring grid points and at neighboring times, to compensate for systematic timing or location bias in cyclone landfall. Also one can test using NWP forecasts of other weather elements (temperature, humidity, wind, etc) as predictors in the DEF algorithm for precipitation, and can explore the use of different GEP parameter settings at different stations. The work of Cannon (2008) and others can be used to explore whether GEP can be used in two stages: first to classify the forecast precipitation (extreme, normal, or none), and then to use different DEF algorithms for the extreme vs. normal events.

In addition to finding DEFs, one can apply GEP to each ensemble member individually strictly as a bias-correction method, yielding a set of ensemble members that can be used for probabilistic forecasts. GEP DEFs can be compared with additional regression and ensemble-averaging approaches, such as Bayesian Model Averaging. The weaker ensemble members can be investigated to determine what aspects of the NWP dynamics or physics were inadequate, with the aim of suggesting improvements to the NWP codes. All of this work has the ultimate goal of improving the skill of weather forecasts over complex terrain.

6.2 Short term electric load forecasting using gene expression programming

Two different methods for electric load forecasting were tested in Chapter 3 and Chapter 4.

6.2.1 Method in chapter 3

To estimate electric load via a nonlinear combination of weather, calendar and load data, we used GEP as regression tool. For western Canada, the typical electric load curve has relative minima during nighttime and midday, and relative maxima in the morning and evening. We estimate the load for each of these four extrema by first removing the trend, annual cycle, and semi-annual cycle. Then, we use GEP to fit the remaining load signal (separately for each extreme) as a function of air temperature, wind speed, precipitation, humidity, day-of-the-week, and other meteorological and

calendar variables. This is done via a multi-year training set of load observations and weather data.

Once these best-fit algorithms are found, we use them to forecast load extrema for subsequent days. We then use a Bézier curve to interpolate between the extrema to get hourly load forecasts. This method is verified against independent data for a year of daily load forecasts.

6.2.2 Findings from chapter 3

Although the four-point short-term forecasting looked promising, operational forecasting for year 2010 drew our attentions to its pitfalls including: 1) the shape of daily load signal varies month to month; 2) the interpolation scheme may cause a huge error for small shifts in forecast time; 3) updating each model is not an easy task because of: a) using too many variables; and b) using the sum of too many fixed harmonics. These shortcomings lead us to test a simpler alternative electrical load forecasting model in Chapter 4.

6.2.3 Method in chapter 4

A two-stage (predictor-corrector) approach is used, where the first stage uses a single regression model that applies weather and calendar data of the previous hour to predict load for any hour of the day, and the second stage applies different corrections every hour, based on high correlation of today's load with yesterday's load for any given hour. By excluding the day-before variables, the two-stage method reduces the total number of variables in first-stage regression and gives better results. Simpler weather data is used− only CYVR temperature.

We used seven years of hourly temperature and electrical load data for British Columbia in western Canada are used to compare two statistical methods, artificial neural networks (ANN) and gene expression programming (GEP) (with same design and comparable conditions), to produce hour-ahead load forecasts. Two linear control methods (persistence; multiple linear regression) are used for verification.

6.2.4 Findings from chapter 4

We find that a two-stage (predictor-corrector) statistical process gives good robust load forecasts. Nonlinear statistical methods GEP and ANN work equally well for the first stage, and they capture most of the variance in the one-hour-ahead load signal. The remaining errors seem to have a repeating pattern from one day to the next. This allows one to use the error from 24 hours ago as a bias corrector in a second statistical stage to improve the forecast for the present hour. For this second stage, almost any nonlinear or linear method (including 24-hour persistence) works well. This two-stage method reduces the forecast mean-absolute error to about 0.6% of the total electric

load. The hour-ahead forecasts can be used iteratively to forecast more hours ahead, with the error increasing to about 3.5% of the total load after 24 hours of iteration.

6.2.5 Comparison of results from chapters 3 and 4

The GEP genome in Chapter 3 was much longer and more complex than in Chapter 4. Also, more weather variables were used from four cities as predictors for GEP in Chapter 3, compared to the one weather variable (air temperature) at one city (Vancouver) in Chapter 4.

In Chapter 3 a load forecast was made for all hours (1 to 24) ahead of the current time. In contrast, chapter 4 focused mostly on one-hour-ahead forecasts, which are recomputed each hour, although it was shown how one-hour-forecasts can be chained together to forecast 24 h or more into the future.

Both load-forecast chapters used a "perfect prog" approach, where observed weather was used as input. This allows the verification to focus on the quality of the load algorithms without being confounded by weather-forecast errors. Both chapters used extensive load-observation data provided by BC Hydro. Also, the short forecast horizons (1 to 24 h) studied in both chapters are relevant to BC Hydro's short-range load-forecast needs.

Insight can be gained by comparing the spread of load errors in both chapters. Figure 3.8 to Figure 3.10 show somewhat broad load spreads with substantial frequencies of error magnitudes of 300 MW and greater. In Chapter 4, in spite of the simpler approach, the tails of the load-error distributions in Figure 4.5 and Figure 4.6 are smaller, with almost all error magnitudes of 200 MW or smaller.

In both chapters, GEP was compared to artificial neural network (ANN) load forecasts. One motivation is that ANNs are very popular tools for load forecasts by electric utility companies, including BC Hydro, which uses the ANNSTLF commercial package. In both chapters we found that GEP had load-forecasts skill comparable to ANN, and thus GEP could be a valuable tool to these utility companies.

6.2.6 Future work

Although using additional weather variables such as dew point temperature, wind speed and weather conditions did not add value to our one-hour-ahead forecasts, it could add value to the critical points in the load curve (Chapter 3). The combination of two load models may improve forecasts. Inspired by the structure of NWP models and their nesting in space, future work could include combining load forecasts from GEP for holidays with ANN for weekends and weekdays, with different lead-time nesting.

6.3 Saturated pseudoadiabats parameterization using gene expression programming

To maximize the utility of clean energy generated by hydroelectricity or wind turbines, better weather forecasts are needed. Severe weather such as thunderstorms can damage wind turbines and cause extreme water inflows into hydroelectric reservoirs. To aid thunderstorm prediction, GEP is used to create a computationally efficient approximation to moist convective lapse rates.

6.3.1 Method

Two non-iterative approximations were presented for saturated pseudoadiabats (also known as moist adiabats). One approximation allows you to determine which moist adiabat passes through a point of known pressure and temperature, such as is drawn on a tephigram, skew-T diagram, or other thermodynamic diagram. The other approximation allows you to determine the air temperature at any pressure along a known moist adiabat. GEP is used to find these approximations.

6.3.2 Findings

The correlation coefficient r between the approximations and the thermodynamic diagram data is over 99.7%. The difference between the non-iterative approximation and the corresponding iterated target values is less than about 0.5°C over a majority of the skew-T subdomain . The mean absolute error is 0.28°C and the root mean squared error is 0.44°C within a thermodynamic domain bounded by $-30 < \theta_w \leq 40$°C, $P > 20$ kPa, and$-60 \leq T \leq 40$°C, where θ_w, P, T are wet-bulb potential temperature, pressure, and air temperature respectively.

Although iteration is not needed when using approximations to determine the thermodynamic state of a rising saturated air parcel, some meteorological applications require iteration nonetheless. Examples are buoyancy or stability calculations. Iteration is required for these cases because the environment surrounding the air parcel also changes as the parcel rises. Nonetheless, approximations allow one to take larger vertical steps such as might correspond to significant levels in a sounding, rather than taking the smaller steps that would have been needed to accurately follow a saturated adiabat using iterative equations.

6.3.3 Future work

The two approximations provide satisfying results as non-iterative solutions. One can perform more GEP evolutions to see if more accurate approximations are possible.

Bibliography

Adami, C., 1998: *Introduction to Artificial Life*. Springer, 374 pp.

Alfares, H. K. and M. Nazeeruddin, 2002: Electric load forecasting: Literature survey and classification of methods. *International Journal of Systems Science*, **33**, 23–34.

Ambaum, M., 2010: *Thermal Physics of the Atmosphere*. Wiley-Blackwell, 239 pp.

Ambaum, M., 2011: Tephigram. URL http://www.met.reading.ac.uk/~sws97mha/Tephigram/.

Amjady, N., 2007: Short-term bus load forecasting of power systems by a new hybrid method. *IEEE Trans. Power Syst.*, **22 (1)**, 333–341.

ATAA, 1942: Pseudo-adiabatic diagram. Air Transport Association of America; Meteorological Committee Chart VII, 1-42 pp.

Aytek, A., M. Asce, and M. Alp, 2008: An application of artificial intelligence for rainfall-runoff modelling. *J. Earth System Science*, **117 (2)**, 145–155, URL http://dx.doi.org/10.1007/s12040-008-0005-2, 10.1007/s12040-008-0005-2.

Bakhshaii, A. and R. Stull, 2009: Deterministic ensemble forecasts using gene-expression programming. *Wea. Forecasting*, **24**, 1431–1451, doi:10.1175/2009WAF2222192.1.

Bakhshaii, A. and R. Stull, 2011: Gene-expression programming−an electrical-load forecast alternative to artificial neural networks. *91st American Meteorological Society Annual Meeting*, Seattle,WA.

Bakhshaii, A. and R. Stull, 2012: Electric load forecasting for W. Canada: A comparison of two nonlinear methods. *Atmosphere-Ocean*, accepted.

Bashir, Z. and M. El-Hawary, 2009: Applying wavelets to short-term load forecasting using pso-based neural networks. *IEEE Trans. Power Syst.*, **24 (1)**, 20–27.

BC Hydro, 2012: *BC Hydro's System*. URL http://www.bchydro.com/energy_in_bc/our_system.html.

Beccali, M., M. Cellura, V. Lo Brano, and A. Marvuglia, 2004: Forecasting daily urban electric load profiles using artificial neural networks. *Energy Convers. Manage.*, **45**, 2879–2900.

Benoit, R. M., M. Desgagne, P. Pellerin, S. Pellerin, and Y. Chartier, 1997: The canadian MC2: A semi-lagrangian, semi-implicit wideband atmospheric model suited for finescale process studies and simulation. *Mon. Wea. Rev.*, **125**, 2383–2415.

Bohren, C. and B. Albrecht, 1998: *Atmospheric Thermodynamics*. Oxford University Pressl, 402 pp.

Bolton, D., 1980: The computation of equivalent potential temperature. *Mon. Wea. Rev.*, **108**, 1046–1053.

Bozic, S. M., 1994: *Digital and Kalman filtering: an introduction to discrete-time filtering and optimum linear estimation*. 2d ed., Halsted Press, 160 pp.

Bremermann, H., 1962: *Optimization through evolution and recombination*, 93–106. Spartan Books.

Cannon, A., 2008: Probabilistic multisite precipitation downscaling by an expanded bernoulli-gamma density network. *J. Hydrometeor.*, **9**, 1284–1300, doi:0:1175/2008JHM960.1.

Chan, Z. S. H., H. W. Ngan, A. B. Rad, A. K. David, and N. Kasabov, 2006: Short-term ann load forecasting from limited data using generalization learning strategies. *Neurocomputing*, **70 (1-3)**, 409–419.

Charles, M. and C. B.A., 2009: Verification of extratropical cyclones within the ncep operational models, part ii: The short-range ensemble forecast system. *Mon. Wea. Rev.*, **24**, 1191–1213.

Chen, B.-J., M.-W. Chang, and C.-J. lin, 2004: Load forecasting using support vector machines: a study on eunite competition 2001. *IEEE Trans. Power Syst.*, **19 (4)**, 1821–1830.

Chen, H., C. Canizares, and A. Singh, 2001: Ann-based short-term load forecasting in electricity markets. *IEEE Power Engineering Society Transmission and Distribution Conf.*, Vol. 2, 411–415.

Cheng, W. and W. Steenburgh, 2007: Strengths and weaknesses of mos, running-mean bias removal, and kalman filter techniques for improving model forecasts over the W. United States. *Wea. Forecasting*, **22**, 1304–1318.

Cramer, N., 1985: A representation for the adaptive generation of simple sequential programs. *Proceedings of the First International Conf. on Genetic Algorithms and Their Applications*, 183–187, edited by Grefenstette JJ. Erlbaum.

da Silva, A. and L. Moulin, 2000: Confidence intervals for neural network based short-term load forecasting. *IEEE Trans. Power Syst.*, **15 (4)**, 1191–1196.

Darwin, C., 1859: *On the Origin of Species by Means of Natural Selection*. John Murray, London, 502 pp.

Delle Monache, L., T. Nipen, X. Deng, Y. Zhou, and R. Stull, 2006: Ozone ensemble forecasts: 2. a kalman filter predictor bias correction. *J. Geophys. Res.*, **111**, doi:10.1029/2005JD006311.

Dey, D., P. Muller, and D. Sinha, 1998: *Practical nonparametric and semiparametric Bayesian statistics*. Springer, 369 pp.

Donoho, D. L. and I. M. Johnstone, 1995: Adapting to unknown smoothness via wavelet shrinkage. *J. of the American Statistical Association*, **90**, 1200–1224.

Donoho, D. L., I. M. Johnstone, G. Kerkyacharian, and D. PicardSource, 1995: Wavelet shrinkage: Asymptopia. *J. of the Royal Statistical Society*, **57**, 301–369.

Drezga, I. and S. Rahman, 1998: Input variable selection for ANN-based short-term load forecasting. *IEEE Trans. Power Syst.*, **13 (4)**, 1238–1244.

EC, 1976: Tephigram. Environment Canada, Atmospheric Environment Service.

Eckel, F. and C. Mass, 2005: Aspects of effective mesoscale, short-range ensemble forecasting. *Wea. Forecasting*, **20**, 328–350.

ECMWF, 2007: Verification and applications of ensemble forecasts. *WG 3. A report from Working Group 3 (chaired by Ken Mylne) of the Workshop on Ensemble Prediction*, 6.

Eiben, A. and J. Smith, 2003: *Introduction to Evolutionary Computing*. Springer, 307 pp.

Emanuel, K., 1994: *Atmospheric Convection*. Oxford University Press, 580 pp.

EPRI, 2009: Artificial neural network short-term load forecaster 5.2. http://my.epri.com/portal/server.pt?Abstract_id=000000000001018874.

Eubank, R., 1999: *Nonparametric Regression and Spline Smoothing*. 2d ed., Marcel Dekker, 337 pp.

Fan, S. and L. Chen, 2006: Short-term load forecasting based on an adaptive hybrid method. *IEEE Trans. Power Syst.*, **21 (1)**, 392–401.

Farlow, S., 1984: *Self-organizing Methods in Modeling: GMDH Type Algorithms*. Marcel Dekker, 350 pp.

Feinberg, A. and D. Genethliou, 2005: *Applied Mathematics for Restructured Electric Power Systems: Optimization, Control and Computational Intelligence, Power Electronics and Power Systems*, chap. Load Forecasting, 269–285. Springer.

Feinberg, A., D. Genethliou, and T. Hauagos, 2003: Statistical load modeling. *7th IASTED International Multi Conference on Power and Energy Systems*, 88–91.

Ferreira, C., 2001: Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems*, **13 (2)**, 87–129.

Ferreira, C., 2006: *Gene Expression Programming; Mathematical Modelling by an Artificial Intelligence*. 2d ed., Springer, 478 pp.

Ferreira, C., 2008: *Gene Expression Programming*. URL http://www.gepsoft.com.

Fildes, R., K. Nikolopoulos, S. F. Crone, and A. A. Syntetos, 2008: Forecasting and operational research: a review. *J. Oper. Res. Soc.*, **59 (9)**, 1150–1172.

Fogel, L., A. Owens, and M. Walsh, 1966: *Artificial Intelligence through Simulated Evolution*. NY: John Wiley.

Friedman, J., 1991: Multivariate adaptive regression splines. *The Annals of Statistics*, **19**, 1–141.

Friedman, J. and W. Stuetzle, 1981: Projection pursuit regression. *J. of the American Statistical Association*, **76**, 817–823.

Gel, Y., 2007: Comparative analysis of the local obs.-based (LOB) method and the non-parametric regression-based method for gridded bias correction in mesoscale weather forecasting. *Wea. Forecasting*, **22**, 1243–1256.

Goldberg, D., 1989: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Grell, G., J. Dudhia, and D. Stauffer, 1994: A description of the fifth-generation Penn State/NCAR mesoscale model (MM5). Technical Report TN-398+STR, National Center for Atmospheric Research.

Grubisic, V., R. Vellore, and A. Huggins, 2005: Quantitative precipitation forecasting of wintertime storms in the Sierra Nevada: sensitivity to the microphysical parameterization and horizontal resolution. *Mon. Wea. Rev.*, **133**, 2834–2859.

Hacker, J. and D. Rife, 2007: A practical approach to sequential estimation of systematic error on near-surface mesoscale grids. *Wea. Forecasting*, **22**, 1257–1273.

Hamill, T. and S. Colucci, 1997: Verification of eta-rsm short-range ensemble forecasts. *Mon. Wea. Rev.*, **125**, 1312–1327.

Hamill, T., S. Mullen, C. Snyder, Z. Toth, and D. Baumhefner, 2000: Ensemble forecasting in the short to medium range: Report from a workshop. *Bull. Amer. Meteor. Soc.*, **81**, 2653–2664.

Hand, D., 1997: *Construction and Assessment of Classification Rules*. NY John Wiley and Sons, 214 pp.

Hansen, B., 2007: A fuzzy-logic based analog forecasting system for ceiling and visibility. *Wea. Forecasting*, **22**, 1319–1330.

Härdle, W., 1990: *Applied Nonparametric Regression*. Cambridge Univ. Press., 329 pp.

Hess, S., 1959: *Introduction to Theoretical Meteorology*. Henry Holt and Co, 362 pp.

Hinojosa, V. H. and A. Hoese, 2010: Short-term load forecasting using fuzzy inductive reasoning and evolutionary algorithms. *IEEE Trans. Power Syst.*, **25 (1)**, 565–574.

Hippert, H. and C. Pedreira, 2004: Estimating temperature profiles for short-term load forecasting: neural networks compared to linear models. *Generation, Transmission and Distribution, IEE Proceedings-*, **151 (4)**, 543–547, doi:10.1049/ip-gtd:20040491.

Hippert, H., C. Pedreira, and R. Souza, 2001: Neural networks for short-term load forecasting: A review and evaluation. *IEEE Trans. Power Syst.*, **16**, 44–55.

Holland, J., 1975: *Adaptation in Natural and Artificial Systems*. Univ. Michigan Press, reprinted in 1992 by MIT Press.

Hsieh, W., 2009: *Machine Learning Methods in the Environmental Sciences:Neural Networks and Kernels.* 2d ed., Cambridge Univ. Press., 364 pp.

Huang, C. M. and H. T. Yang, 2001: Evolving wavelet-based networks for short-term load forecasting. *Generation, Transmission and Distribution, IEE Proceedings-*, **148 (3)**, 222–228.

Jones, M., B. Colle, and J. Tongue, 2007: Evaluation of a mesoscale short-range ensemble forecast system over the northeast United States. *Wea. Forecasting*, **22**, 36–55.

Kalnay, E., 2003: *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge Univ. Press, 341 pp.

Khotanzad, A., R. Afkhami-Rohani, and D. Maratukulam, 1998: ANNSTLF-Artificial neural network short-term load forecastergeneration three. *IEEE Trans. Power Syst.*, **13**, 1413–1422.

Koza, J., 1992: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 840 pp.

Koza, J., 1994: *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, 768 pp.

Koza, J., F. Bennett III, D. Andre, and M. Keane, 1999: *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, 1154 pp.

Koza, J., M. Keane, M. Streeter, W. Mydlowec, J. Yu, and G. Lanza, 2003: *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Springer, 590 pp.

Lee, K., Y. Cha, and J. Park, 1992: Short-term load forecasting using an artificial neural network. *IEEE Trans. Power Syst.*, **7**, 124–132.

Lemaître, O., 2010: *Weather and Climate Risk in the Energy Industry*, chap. Meteorology Climate and Energy, 51–56. Springer.

Lenski, R., C. Ofria, R. Pennock, and C. Adami, 2003: The evolutionary origin of complex features. *Nature*, **423**, 139–144.

Liao, G.-C. and T.-P. Tsao, 2006: Application of a fuzzy neural network combined with a chaos genetic algorithm and simulated annealing to short-term load forecasting. *IEEE Trans. Evolutionary Computation*, **10 (3)**, 330–340.

Lim, T., W. Loh, and Y. Shih, 2000: A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, **40**, 203–229.

Ling, S. H., F. H. F. Leung, H. K. Lam, Y.-S. Lee, and P. K. S. Tam, 2003: A novel genetic-algorithm-based neural network for short-term load forecasting. *Industrial Electronics, IEEE Transactions on*, **50 (4)**, 793–799.

Mao, H., X.-J. Zeng, G. Leng, Y.-J. Zhai, and J. A. Keane, 2009: Short-term and midterm load forecasting using a bilevel optimization model. *IEEE Trans. Power Syst.*, **24 (2)**, 1080–1090.

Marin, F. J., F. Garcia-Lagos, G. Joya, and F. Sandoval, 2002: Global model for short-term load forecasting using artificial neural networks. *Generation, Transmission and Distribution, IEE Proceedings-*, **149 (2)**, 121–125.

McCollor, D. and R. Stull, 2008a: Hydrometeorological accuracy enhancement via postprocessing of numerical weather forecasts in complex terrain. Part I: meteorological evaluation. *Wea. Forecasting*, **23**, 533–556, dOI: 10.1175/2007WAF2006107.1.

McCollor, D. and R. Stull, 2008b: Hydrometeorological accuracy enhancement via postprocessing of numerical weather forecasts in complex terrain. Part II: economic evaluation. *Wea. Forecasting*, **23**, 557–574.

McCollor, D. and R. Stull, 2009: Evaluation of probabilistic medium-range temperature forecasts from the north american ensemble forecast system. *Wea. Forecasting*, **24**, 3–17.

Mitchell, T., 1997: *Machine Learning*. McGraw-Hill, 414 pp.

Mori, H. and N. Kosemura, 2001: Optimal regression tree based rule discovery for short-term load forecasting. *IEEE Power Engineering Society Winter Meeting Transmission and Distribution Conference*, Vol. 2, 421–426.

Musilek, P., E. Pelikan, T. Brabec, and M. Simunek, 2006: Recurrent neural network based gating for natural gas load prediction system. *IEEE International Conference on Neural Networks*, Vancouver, BC, Canada, 3736–3741, 1716612.

Osowski, S. and K. Siwek, 2002: Regularisation of neural networks for improved load forecasting in the power system. *Generation, Transmission and Distribution, IEE Proceedings-*, **149 (3)**, 340–344.

Park, D., M. El-Sharkawi, R. Marks, L. Atlas, and M. Damborg, 1991: Electric load forecasting using an artificial neural network. *IEEE Trans. Power Syst.*, **6**, 442–449.

Peel, M., B. Finlayson, and T. McMahon, 2007: Updated world map of the Köppen-Geiger climate classification. *Hydrol. Earth Syst. Sci.*, **11**, 1633–1644.

Raftery, A., T. Gneiting, F. Balabdaoui, and M. Polakowski, 2005: Using bayesian model averaging to calibrate forecast ensembles. *Mon. Wea. Rev.*, **133**, 1155–1174.

Rechenberg, I., 1965: Cybernetic solution path of an experimental problem. Tech. rep. Royal Aircraft Establishment, Library Translation.

Richardson, D., 2000: Skill and relative economic value of the ecmwf ensemble prediction system. *Quart. J. Roy. Meteor. Soc.*, **126**, 649–667.

Ripley, B., 1996: *Pattern Recognition and Neural Networks*. Cambridge University Press.

Roebber, P., 2010: Seeking consensus: a new approach. *Mon. Wea. Rev.*, **138**, 4402–4415, doi:10.1175/2010MWR3508.1.

Sadat Hosseini, S. and A. Gandomi, 2010: Short-term load forecasting of power systems by gene expression programming. *Neural Computing and Applications*, 1–13, URL http://dx.doi.org/10.1007/s00521-010-0444-y, 10.1007/s00521-010-0444-y.

Schwefel, H.-P., 1965: Kybernetische Evolution als Strategie der Experimentellen Forschungin der Strömungstechnik. Diploma Thesis.

Skamarock, W., J. Klemp, J. Dudhia, D. Gill, D. Barker, W. Wang, and J. Powers, 2005: A description of the advanced research WRF version 2. Technical Report TN-468+STF, NCAR, 88 pp.

Srinivasan, D., S. S. Tan, C. S. Cheng, and E. K. Chan, 1999: Parallel neural network-fuzzy expert system strategy for short-term load forecasting: system implementation and performance evaluation. *IEEE Trans. Power Syst.*, **14 (3)**, 1100–1106.

Stensrud, D. and J. Skindlov, 1996: Grid point predictions of high temperature from a mesoscale model. *Wea. Forecasting*, **11**, 103–110.

Stensrud, D. and N. Yussouf, 2003: Short-range ensemble predictions of 2-m temperature and dewpoint temperature over New England. *Mon. Wea. Rev.*, **131**, 2510–2524.

Stensrud, D. and N. Yussouf, 2005: Bias-corrected short-range ensemble forecasts of near-surface variables. *Meteor. Appl.*, **12**, 217–230.

Stensrud, D. and N. Yussouf, 2007: Reliable probabilistic quantitative precipitation forecasts from a short range ensemble forecasting system. *Wea. Forecasting*, **22**, 3–17.

Stull, R., 2011a: *Meteorology for Scientists and Engineers*. 3d ed., Discount Textbooks, 924 pp.

Stull, R., 2011b: Wet-bulb temperature from relative humidity and air temperature. *J. Appl. Meteor. Climatol.*, **50 (11)**, 2267–2269, URL http://dx.doi.org/10.1175/JAMC-D-11-0143.1.

Taylor, J. and R. Buizza, 2002: Neural network load forecasting with weather ensemble predictions. *IEEE Trans. Power Syst.*, **17**, 626–632.

Turing, A., 1946: Proposed electronic calculator. Technical report, National Physical Laboratory, Teddington.

USGS, 2011: U. S. geological survey. URL http://ga.water.usgs.gov/edu/wuhy.html.

Vaisala, 2010: Datasheet for vaisala radiosonde rs92-sgp. Tech. rep. URL http://www.vaisala.com/Vaisala%20Documents/Brochures%20and%20Datasheets/RS92SGP-Datasheet-B210358EN-E-LoRes.pdf.

Wahlgren, R. V., 2009: SW BC proportion of BC Hydro grid activity. Bc climate zones project, BC Hydro Customer Information Management-Load Analysis, 7 pp.

Wandishin, M., S. Mullen, D. Stensrud, and H. Brooks, 2001: Evaluation of a short-range multi-model ensemble system. *Mon. Wea. Rev.*, **129**, 729–747.

Wilks, D., 2006: *Statistical Methods in the Atmospheric Sciences*. 2d ed., Academic Press/Elsevier, 627 pp., iSBN 0-12-751966-.

Woodcock, F. and C. Engel, 2005: Operational consensus forecasts. *Wea. Forecasting*, **20**, 110–111.

Yuan, H., X. Gao, S. Mullen, S. Sorooshian, J. Du, and H. Juang, 2007: Calibration of probabilistic quantitative precipitation forecasts with an artificial neural network. *Wea. Forecasting*, **22**, 1287–1303.

Yun, Z., Z. Quan, S. Caixin, L. Shaolan, L. Yuming, and S. Yang, 2008: RBF neural network and ANFIS-based short-term load forecasting approach in real-time price environment. *IEEE Trans. Power Syst.*, **23 (3)**, 853–858.

Yussouf, N. and D. Stensrud, 2006: Prediction of near-surface variables at independent locations from a bias-corrected ensemble forecasting system. *Mon. Wea. Rev.*, **134**, 3415–3424.

Yussouf, N. and D. Stensrud, 2007: Bias-corrected short-range ensemble forecasts of near-surface variables during the 2005/06 cool season. *Wea. Forecasting*, **22**, 1274–1286.

Zhu, Y., Z. Toth, R. Wobus, D. Richardson, and K. Mylne, 2002: The economic value of ensemble-based weather forecasts. *Bull. Amer. Meteor. Soc.*, **83**, 73–83.

Appendix A

Gene Expression Programming Details

Although Figure 2.1 and Section 2.2.3 of the Chapter 2 give the basics of Gene-expression Programming (GEP), more details are presented here. Each gene consists of a head (h) domain and tail (t) domain. The head can contain operators, functions, and terminals (predictors). The tail can contain only terminals. During mutations and other modifications to the gene, any character in the head can change to an operator, function, or terminal, but any character in the tail can change only to a terminal. The 80 functions and operators that we allowed to be used during evolution are listed in Appendix D, along with the reduced set of 13 functions that we employed for the Vancouver Airport (YVR) illustrations in the Chapter 2.

To ensure that the tail holds sufficient characters to supply arguments for all the functions and operators in the head domain, the tail length N_t is given by $N_t = N_h(N_{max} - 1) + 1$, where N_h is the number of characters in the head, and N_{max} is the number of arguments in the one function or operator having the most arguments (i.e., the largest arity). For example (Ferreira, 2006), "consider a gene for which the set of functions is $\{Q, *, /, -, +\}$ and the set of terminals is $\{a, b\}$". The symbol Q represents the square-root function, which has only one argument, but all the other operators in this example take two arguments. So the largest arity is $N_{max} = 2$. "If we choose $h = 15$, then $t = 15(2-1)+1 = 16$, and the [total] length of the gene ... is $15+16 = 31$ [characters]."

For example (Ferreira, 2006), suppose that a 31-character gene is:

***b+a-aQab+//+b+babbabbbababbaaa**

where the underlined portion indicates the head. Reading this from left to right defines the corresponding expression tree is Figure A.1.

Note that only the first 8 characters of this 31-character gene is "read", while the last 23 characters are "noncoding" (not used to define the algorithm as given by the expression tree).

But look what happens if there is just one mutation in position 6 of the gene, changing "a" to "+":

***b+a-+Qab+//+b+babbabbbababbaaa**,

then the resulting expression tree is illustrated in Figure A.2. This uses 20 characters from the gene, and spans all of the head plus a portion of the tail.

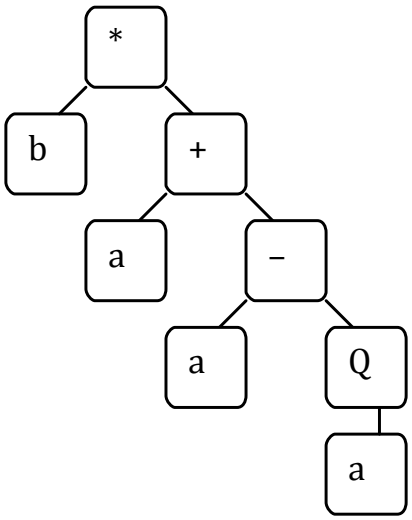


Figure A.1: Expression tree.

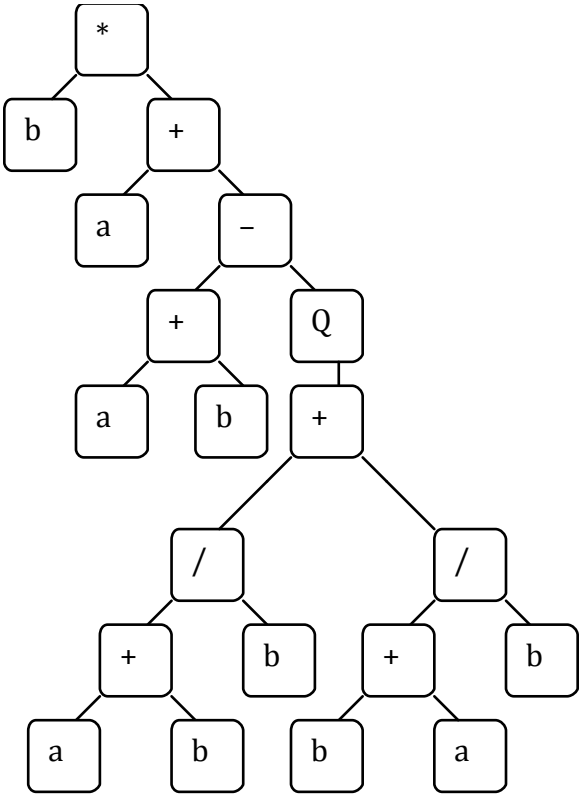


Figure A.2: Modified expression tree.

If you have difficulty constructing the expression tree from the gene, try going in reverse. Namely, as shown in Figure 2.1 of the Chapter 2, read the expression tree like a book: reading each row from the top down, and within any one row read from left to right (namely, do NOT follow the expression tree branches). It is this book-like reading that makes GEP so much more efficient at evolution compared to traditional genetic algorithms and genetic programs.

In the above examples, the coded string of characters is like a genotype (i.e., the genetic information), while the expression tree is like the corresponding phenotype (i.e., the physical body informed by the genotype). Although we have been showing the phenotype as an expression tree, this tree can easily be coded into any of a variety of computer programming languages and scripts. Appendix E shows an example of the MatLab phenotype for one of the actual algorithms evolved to determine precipitation bias for Vancouver Airport. The GEP software allows the user to pick the output language for the algorithm, such as FORTRAN, C, MatLab, etc, which the GEP program automatically produces for the evolved best individual.

The phenotype and genotype together define the individual. During evolution, it is the genotype that is modified by mutation and other processes (Appendix C) that mimic evolutionary processes in biological life. The resulting phenotype (the algorithm) is tested for fitness (how well it verifies compared to meteorological observations based on the training set of data), and used to decide which individuals spawn daughters in the next generation (with or without modification). A population can consist of hundreds to thousands of individuals, each possibly unique and each evolving from generation to generation.

Appendix B

Statistical Measures

The following statistical functions are used to determine fitness f of individual algorithms (Ferreira, 2006). For data index i between 1 and n data points, let E_i be the value estimated by the evolved algorithm, and O_i be the target value. For some of our experiments, $(E,O) =$ (DEF bias-corrected precipitation amount, observed precipitation amount), while for other experiments $(E,O) =$ (DEF precipitation bias, precipitation bias between observation and MC2 4km forecast). Root relative squared error (RRSE) is:

$$\text{RRSE} = \left[\frac{\sum_{i=1}^{n}(E_i - O_i)^2}{\sum_{i=1}^{n}(\bar{O} - O_i)^2}\right]^{1/2}$$

where $\bar{O}$ is the mean target value, averaged over all i. The error in the DEF estimate is "relative" to the error that would have occurred if the estimate was just the average of the observations (namely, the climatology of the training set). From this error, the fitness is:

$$f = 1000(1+\text{RRSE})^{-1}$$

where a perfectly-fit individual has $f = 1000$, and f approaches zero as RRSE increases toward infinity. When parsimony pressure is included, RRSE is modified by adding a very small term that increases as size of the chromosome decreases [see citetFerreira-06 for details].

Mean absolute error (MAE) is:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|E_i - O_i|$$

with fitness $f = 100(1+\text{MAE})^{-1}$

Relative absolute error (RAE) is:

$$\text{RAE} = \frac{\sum_{i=1}^{n}|E_i - O_i|}{\sum_{i=1}^{n}|\bar{O} - O_i|}$$

with fitness

$$f = 100(1+\text{RAE})^{-1}$$

For verification, we verify the estimated precipitation against the paired 24-h accumulated precipitation observation. Even for those algorithms evolved for forecasting precipitation bias, we first convert the estimated bias to a precipitation amount using Eq. 2.1 before verifying against observed precipitation. In addition to MAE, the following statistical measures (Wilks, 2006) are used for verification:

Degree of Mass Balance (DMB) is the ratio of predicted to observed net water mass over the whole verification period (Grubisic et al., 2005):

$$\text{DMB} = \frac{\sum_{i=1}^{n} E_i}{\sum_{i=1}^{n} O_i}$$

For perfect mass balance, DMB = 1.

Mean Error (ME) is:

$$\text{ME} = \frac{1}{n}\sum_{i=1}^{n} E_i - \frac{1}{n}\sum_{i=1}^{n} O_i = \bar{E} - \bar{O}$$

where zero mean error is best. The overbar denotes the average over all data indices.

Root Mean Squared Error (RMSE):

$$\text{RMSE} = \left[\frac{1}{n}\sum_{i=1}^{n}(E_i - O_i)^2\right]^{1/2}$$

where zero error is best. This error measure gives more weight to the outliers than does MAE.

Pearson correlation coefficient (r):

$$r = \frac{\left\{\sum_{i=1}^{n}(E_i-\bar{E})(O_i-\bar{O})\right\}}{\left\{\sum_{i=1}^{n}(E_i-\bar{E})\cdot\sum_{i=1}^{n}(O_i-\bar{O})\right\}^{1/2}}$$

where $r = 1$ is indicates that the estimate and the observation vary together perfectly.

Appendix C

GEP Chromosome Modification Methods

Designed to mimic biological modification of chromosomes, the modification methods of the GEP chromosome include (Ferreira, 2006):

- mutation −replacement of a randomly chosen character in the chromosome with either an operator or function randomly chosen from a set of operators, a terminal (i.e., an ensemble member), or a random constant;
- inversion −reversing character order in a substring, where the start and end positions of the substring in the head of the gene are determined randomly;
- transposition −copying a randomly selected substring (called an "insertion sequence" or "IS"), and inserting that copy at a randomly chosen location in the same chromosome. Three variants of transposition are used: (1) transposition of IS elements into the head region of a gene; (2) transposition of IS elements to the front (root) of the gene (called a "root insertion sequence" or "RIS" transposition; and (3) transposition of whole genes.
- recombination −random selection of two individuals to swap substrings at random locations. Three variants: (1) one-point; (2) two-point; and (3) whole gene recombination.

The first three modification methods are asexual, while the last is sexual because it involves the sharing of genetic information between different individuals. Computational evolution is accelerated relative to biological evolution by applying these modifications at rates ranging from 0.04 for mutation (e.g., 4 mutations per 100-character chromosome per generation) to 0.4 for one-point recombination. These computational modifications are patterned after observed biological modifications that are known to have been important in driving the evolution of life.

Appendix D

Functions Available to GEP for DEF Symbolic Regression for Precipitation Forecasting

The set of functions from which GEP randomly draws during creation or mutation of algorithms is filled with the weighted number of copies of each function listed below. "Arity" is the number of arguments for each function, which is important for determining the tail size of each gene. (x, y, z) and (a, b, c, d) are dummy arguments used by the function in the order listed. These functions were manually chosen from 279 functions available in the GEP software package.

Function	Symbol	Weight	Arity
Basic Arithmetic Functions:			
Addition: (x+y)	+	5	2
Subtraction: (x-y)	-	4	2
Multiplication: (x*y)	*	4	2
Division: (x/y)	/	1	2
Exponential and Power Functions:			
Square root: ($x^{1/2}$)	Sqrt	2	1
Exponential: (e^x)	Exp	2	1
x to the power of 2: (x^2)	X2	2	1
x to the power of 3: (x^3)	X3	2	1
Cube root: ($x^{1/3}$)	3Rt	1	1
Power: (x^y)	Pow	3	2
Gaussian: $\exp^{-2}$)	Gau	1	1
Rounding and Other Simple Functions:			
Round down to integer: floor(x)	Floor	3	1

Function	Symbol	Weight	Arity
Floating-point remainder: mod(x)	Mod	2	2
Inverse: 1/x	Inv	1	1
Negation: -x	Neg	1	1
No operation: x	Nop	1	1
Logarithmic functions:			
Natural logarithm: ln(x)	Ln	1	1
Trigonometric Functions (for x in radians):			
Sine: sin(x)	Sin	1	1
Cosine: cos(x)	Cos	1	1
Tangent : tan(x)	Tan	2	1
Inverse Trig Functions:			
Arctangent: atan(x)	Atan	1	1
Arccosecant : acsc(x)	Acsc	1	1
Arccotangent : acot(x)	Acot	1	1
Hyperbolic Functions:			
Hyperbolic tangent: tanh(x)	Tanh	1	1
Hyperbolic cosecant: csch(x)	Csch	1	1
Hyperbolic secant: sech(x)	Sech	1	1
Hyperbolic cotangent: coth(x)	Coth	2	1
Inverse Hyperbolic Functions:			
Inverse hyperbolic cosine	Acosh	1	1
Inverse hyperbolic tangent	Atanh	1	1
Inverse hyperbolic cosecant	Acsch	1	1
Minimum, Maximum, and Average Functions;			
Maximum of 2 inputs: max(x,y)	Max2	1	2
Minimum of 2 inputs: min(x,y)	Min2	1	2
Minimum of 3 inputs: min(x,y,z)	Min3	1	3
Average of 2 inputs: (a+b)/2	Avg2	1	2
Average of 4 inputs: (a+b+c+d)/4	Avg4	1	4
Constant Functions:			
Zero: 0(x) = 0	Zero	2	1

Function	Symbol	Weight	Arity
One: 1(x) = 1	One	2	1
Zero: 0(x,y) = 0	Zero2	1	2
One: 1(x,y) = 1	One2	1	2
Comparison Functions:			
if x $<$0 OR y $<$ 0, then 1, else 0	OR1	2	2
if x $\geq$ 0 OR y $\geq$ 0, then 1, else 0	OR2	1	2
if x $\leq$ 0 OR y $\leq$ 0, then 1, else 0	OR3	2	2
if x $<$1 OR y $<$ 1, then 1, else 0	OR4	4	2
if x$\geq$1 OR y $\geq$ 1, then 1, else 0	OR5	1	2
if x $\leq$ 1 OR y $\leq$ 1, then 1, else 0	OR6	1	2
if x $<$ 0 AND y $<$ 0, then 1, else 0	AND1	1	2
if x $\geq$ 0 AND y $\geq$ 0, then 1, else 0	AND2	1	2
if x $\leq$ 0 AND y $\leq$ 0, then 1, else 0	AND3	2	2
if x $<$ 1 AND y $<$ 1, then 1, else 0	AND4	1	2
if x $\geq$ 1 AND y $\geq$ 1, then 1, else 0	AND5	1	2
if x $\leq$ 1 AND y $\leq$1, then 1, else 0	AND6	1	2
if x $<$ y, then x, else y	LT2A	3	2
if x $>$ y, then x, else y	GT2A	2	2
if x $\leq$ y, then x, else y	LOE2A	1	2
if x $\geq$ y, then x, else y	GOE2A	2	2
if x = y, then x, else y	ET2A	2	2
if x $\neq$ y, then x, else y	NET2A	1	2
if x $<$ y, then 1, else 0	LT2B	2	2
if x $>$ y, then 1, else 0	GT2B	1	2
if x $\leq$ y, then 1, else 0	LOE2B	2	2
if x $\geq$ y, then 1, else 0	GOE2B	2	2
if x = y, then 1, else 0	ET2B	1	2
if x $\neq$ y, then 1, else 0	NET2B	1	2
if x $<$ y, then (x+y), else (x-y)	LT2C	1	2
if x $>$ y, then (x+y), else (x-y)	GT2C	1	2
if x $\leq$ y, then (x+y), else (x-y)	LOE2C	1	2
if x $\geq$ y, then (x+y), else (x-y)	GOE2C	2	2

Function	Symbol	Weight	Arity
if x = y, then (x+y), else (x-y)	ET2C	2	2
if x $\neq$ y, then (x+y), else (x-y)	NET2C	1	2
if x $<$ y, then (x*y), else (x/y)	LT2D	2	2
if x = y, then (x*y), else (x/y)	ET2D	4	2
if x $\neq$ y, then (x*y), else (x/y)	NET2D	1	2
if x $\leq$ y, then (x+y), else (x*y)	LOE2E	1	2
if x $\neq$ y, then (x+y), else (x*y)	NET2E	1	2
if x $<$ y, then (x+y), else sin(x*y)	LT2F	1	2
if x = y, then (x+y), else sin(x*y)	ET2F	1	2
if x $\neq$ y, then (x+y), else sin(x*y)	NET2F	1	2
if x $\leq$ y, then (x+y), else atan(x*y)	LOE2G	1	2
if x $\neq$ y, then (x+y), else atan(x*y)	NET	2	2

Table D.1: The set of 80 functions made available for all "final setting" runs at all stations.

Function	Symbol	Weight	Arity
Addition	+	4	2
Subtraction	−	4	2
Multiplication	*	4	2
Division	/	1	2
Square root	Sqrt	1	1
Exponential	Exp	1	1
Natural logarithm	Ln	1	1
x to the power of 2	X2	1	1
x to the power of 3	X3	1	1
Cube root	3Rt	1	1
Sine	Sin	1	1
Cosine	Cos	1	1
Arctangent	Atan	1	1

Table D.2: Reduced set of 13 functions used to compare different parameter settings (Table 2.3) for Vancouver Airport

Appendix E

Sample MATLAB Algorithm Output from GEP for Precipitation Forecasts at Vancouver Airport

```
%-------------------------------------------------------------------
% Code generated by GeneXproTools 4.0 on 26/03/2008 1:23:08 PM
% Training Samples:  107
% Testing Samples:   51
% Fitness Function:  RRSE
% Training Fitness:  714.678659643746
% Training R-square: 0.841252910884905
% Testing Fitness:   705.930293551433
% Testing R-square:  0.866029597576304
%-------------------------------------------------------------------

function result = gepModelMC2_B4(d)

G1C0 = 5.990876;          G1C1 = -1.094085;
G2C0 = -5.673554;         G2C1 = 9.717987;
G3C0 = -3.039306;         G3C1 = 7.360748;
G4C0 = 2.495819;          G4C1 = 0.265594;
G5C0 = -1.094085;         G5C1 = -9.883149;
G6C0 = 2.437042;          G6C1 = -3.355499;
G7C0 = -8.08081;          G7C1 = -7.842102;

MC2_108km = 1;  MC2_12km = 2;   MC2_36km = 3;   MC2_4km = 4;
MM5_108km = 5;  MM5_12km = 6;   MM5_36km = 7;   MM5_4km = 8;
WRF_108km = 9;  WRF_12km = 10;  WRF_36km = 11;

varTemp = 0.0;

varTemp = gepNET2E(gepOR4((gepGT2B(gepLT2B(G1C1,d(MC2_4km)),tanh(d(WRF_12km)))*
gepET2C(d(MC2_36km),G1C0)),d(MM5_36km)),gepNET2E(tanh(d(MC2_12km)),
gepOR1(G1C1,d(WRF_36km)))-gepOR6(d(MM5_12km),
gepLT2A(d(WRF_12km),d(MM5_108km)))));
```

```
varTemp = varTemp + (gepLOE2B(d(MC2_36km),floor((d(WRF_12km)*
gepOR3((gepLOE2G((G2C1^G2C1),gepLOE2B(G2C1,d(MM5_36km)))),
gepOR3(gepGau(gepET2A(G2C0,d(WRF_12km))),(G2C1^3))))))^2);

varTemp = varTemp + atan((gepLT2F(d(MC2_36km),d(MC2_108km))-
gepOR4(gepGOE2B(acot(gepLOE2G(gepOR5(d(MM5_12km),G3C1),
gepLT2B(d(WRF_108km),d(WRF_36km)))),d(MC2_108km)),exp((0.0)))));

varTemp = varTemp + (-(gepNET2C(gepGT2B(gepLT2F(gepET2A(gepAND5(d(MM5_4km),G4C0),
(d(WRF_12km)+d(MC2_108km))),gepLOE2E(atan(d(MM5_36km)),d(MC2_4km))),
(gepAND3(d(MM5_108km),d(MC2_108km))+
gepGT2C(d(MM5_4km),d(MC2_12km)))),gepNET2G(G4C0,d(MC2_4km)))));

varTemp = varTemp + atan((floor(gepLOE2G(gepMin3(gepGOE2B(d(WRF_108km),d(WRF_108km)),
(d(MM5_108km)-d(MM5_12km)),G5C1),d(MC2_108km)))*
gepGT2C(gepET2C(gepAND3(G5C0,d(MC2_12km)),d(MC2_12km)),
(gepOR4(d(WRF_12km),d(MM5_12km))+tan(d(WRF_36km))))));

varTemp = varTemp + sech(((-((gepLT2D(gepOR6(((gepOR1(G6C0,d(WRF_12km))+
gepLT2F(d(MC2_12km),d(WRF_12km)))/2),atan(d(MM5_36km))),
sech(d(MC2_36km)))^2)))*gepLT2C(G6C0,d(MC2_4km))));

varTemp = varTemp + gepET2A(sqrt(gepET2B(gepOR4(d(MC2_36km),gepOR4(tan(d(MC2_108km)),
gepNET2E(d(MC2_12km),d(WRF_108km)))),acsch(gepOR2(coth(G7C0),d(MC2_12km))))),
gepMin3(d(WRF_108km),d(MC2_4km),d(MM5_108km)));

result = varTemp;


function result = gepMin3(x, y, z)
result = min(min(x,y),z);

function result = gepOR1(x, y)
if ((x < 0) | (y < 0)), result = 1; else   result = 0; end

function result = gepOR2(x, y)
if ((x >= 0) | (y >= 0)), result = 1; else result = 0; end

function result = gepOR3(x, y)
if ((x <= 0) | (y <= 0)), result = 1;  else  result = 0; end

function result = gepOR4(x, y)
if ((x < 1) | (y < 1)), result = 1;  else  result = 0; end

function result = gepOR5(x, y)
if ((x >= 1) | (y >= 1)),  result = 1;  else result = 0; end

function result = gepOR6(x, y)
if ((x <= 1) | (y <= 1)), result = 1; else result = 0; end
```

```
function result = gepAND3(x, y)
if ((x <= 0) & (y <= 0)), result = 1; else result = 0; end

function result = gepAND5(x, y)
if ((x >= 1) & (y >= 1)), result = 1; else result = 0; end

function result = gepLT2A(x, y)
if (x < y), result = x; else result = y; end

function result = gepET2A(x, y)
if (x == y), result = x; else result = y; end

function result = gepLT2B(x, y)
if (x < y), result = 1; else result = 0; end

function result = gepGT2B(x, y)
if (x > y), result = 1; else result = 0; end

function result = gepLOE2B(x, y)
if (x <= y), result = 1; else result = 0; end

function result = gepGOE2B(x, y)
if (x >= y), result = 1; else result = 0; end

function result = gepET2B(x, y)
if (x == y), result = 1; else result = 0; end

function result = gepLT2C(x, y)
if (x < y), result = (x+y); else result = (x-y); end

function result = gepGT2C(x, y)
if (x > y), result = (x+y); else result = (x-y); end

function result = gepET2C(x, y)
if (x == y), result = (x+y); else result = (x-y); end

function result = gepNET2C(x, y)
if (x ~= y), result = (x+y); else result = (x-y); end

function result = gepGau(x)
result = exp(-(x^2));

function result = gepLT2D(x, y)
if (x < y), result = (x*y); else result = (x/y); end

function result = gepLOE2E(x, y)
if (x <= y), result = (x+y); else result = (x*y); end

function result = gepNET2E(x, y)
if (x ~= y), result = (x+y); else result = (x*y); end
```

```
function result = gepLT2F(x, y)
if (x < y), result = (x+y); else result = sin(x*y); end

function result = gepLOE2G(x, y)
if (x <= y), result = (x+y); else result = atan(x*y); end

function result = gepNET2G(x, y)
if (x ~= y), result = (x+y); else result = atan(x*y); end
```

Appendix F

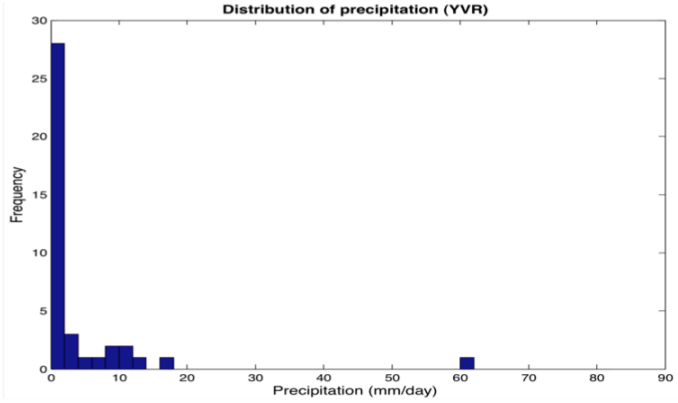
Precipitation Verification Comparison

Forecast errors for the whole scoring set of dates are compared in Figure F.1−Figure F.4, where each figure is for a different weather station (see Table 2.1 in Chapter 2 for station identifiers). A sample of 4 out of the total 24 weather stations are shown. In each figure are three parts (a - c).

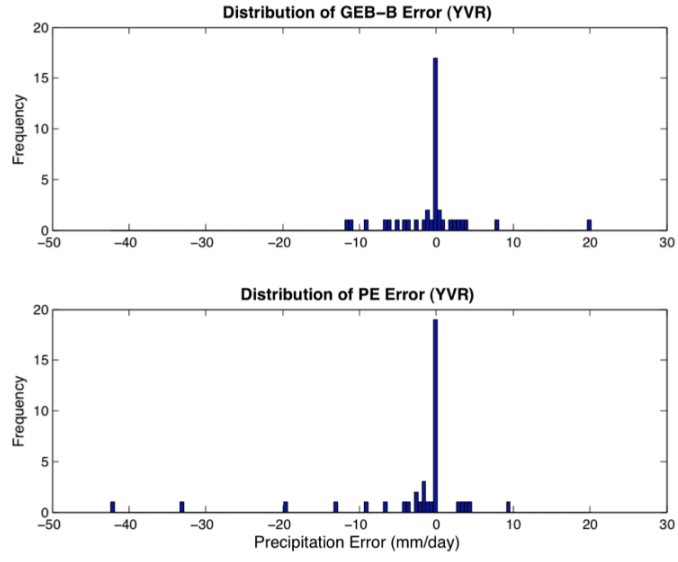
Parts (a) show the usual asymmetric distribution for observed 24-h accumulated precipitation, with many dry days (zero precipitation).

Parts (b) show the spread of errors for two GEP, and PE methods, where the peak near zero means that many of the postprocessed forecasts had near zero error (although many might have been for dry days).

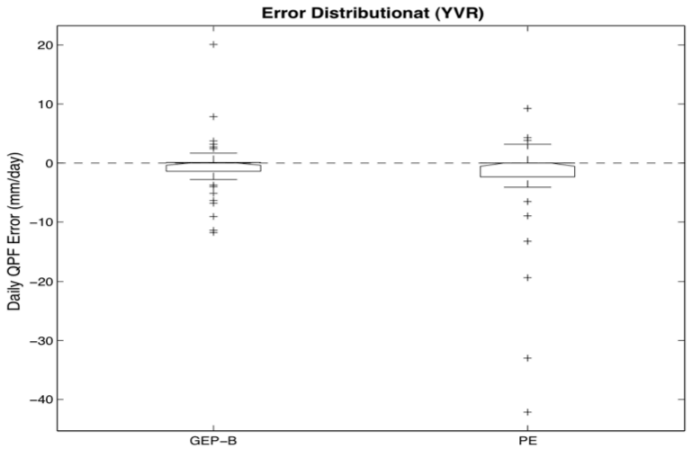
Parts (c) show percentile spreads, where the horizontal box lines are at the lower quartile, median, and upper quartile; whiskers reach the most extreme values within 1.5 times the interquartile range from the ends of the box; and "+" signs mark outliers. The overlap of the interquartile ranges (the boxes) from all three methods show that they are very similar to each other in their ability to forecast observed precipitation, and that it is difficult to discriminate between them. From the outliers we can infer that there are a small number of precipitation events that can be under- or over-forecast by 50 mm/day or more. The existence of large outliers suggests that influences such as the Pacific data void are contributing to random errors that cannot be eliminated by any of these postprocessing methods. This is unfortunate, because the outliers are often associated with heavy precipitation events that can cause floods or landslides.



(a)

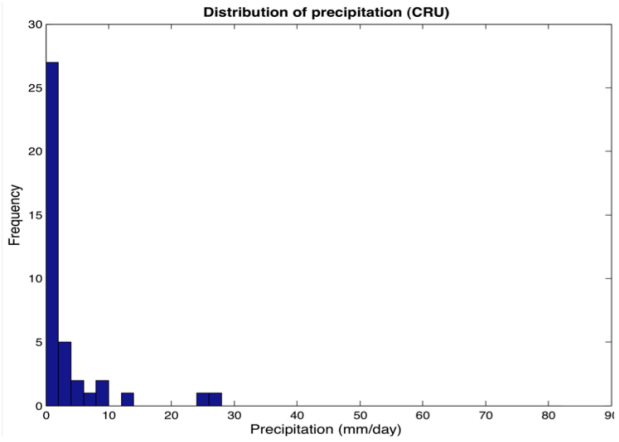


(b)

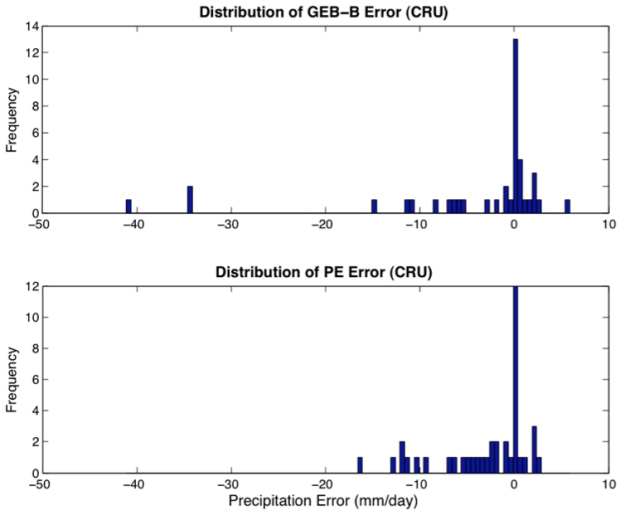


(c)

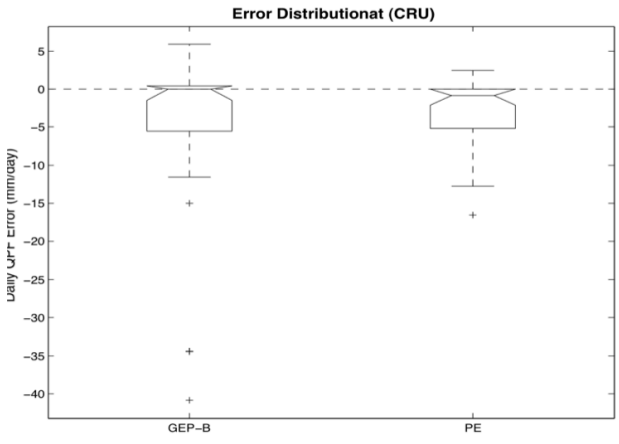
Figure F.1: Verification data for Vancouver Airport (YVR)



(a)

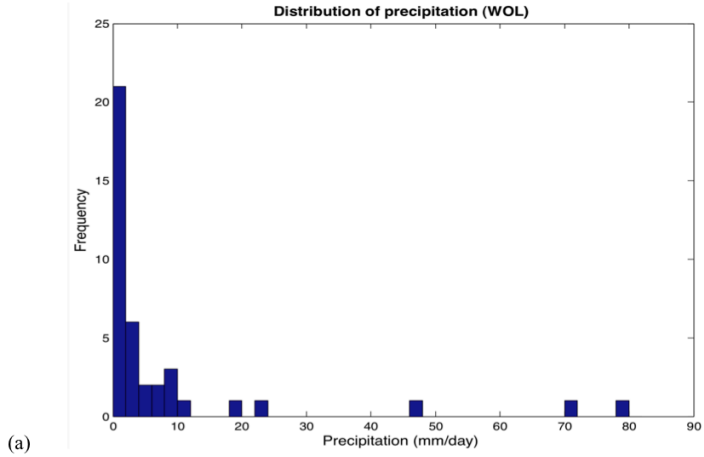


(b)

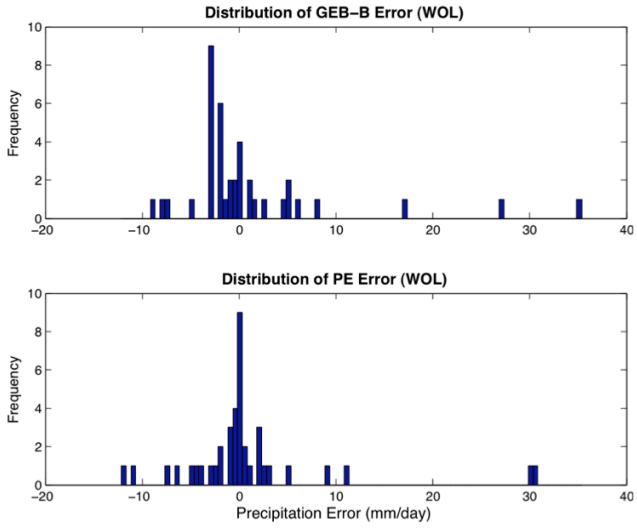


(c)

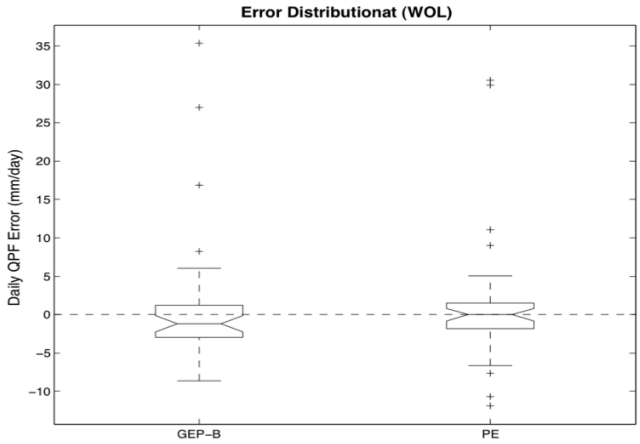
Figure F.2: Same as Figure F.1, but for Cruickshank River (CRU).



(a)

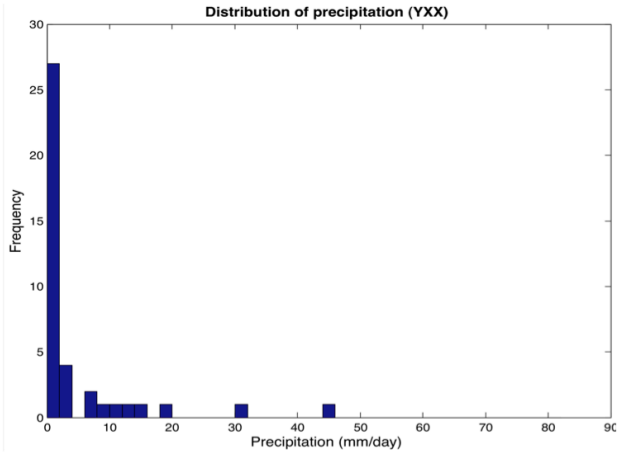


(b)

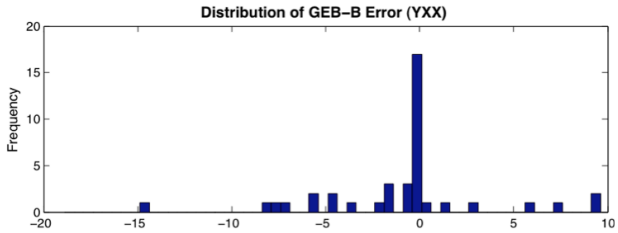


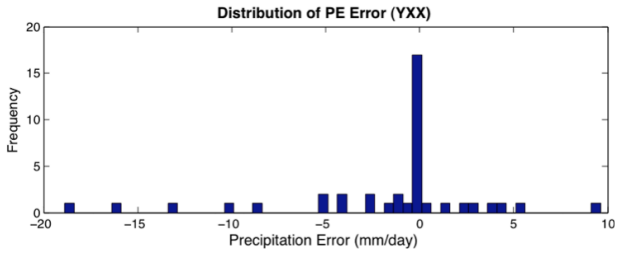
(c)

Figure F.3: Same as Figure F.1, but for Wolf River Upper (WOL).

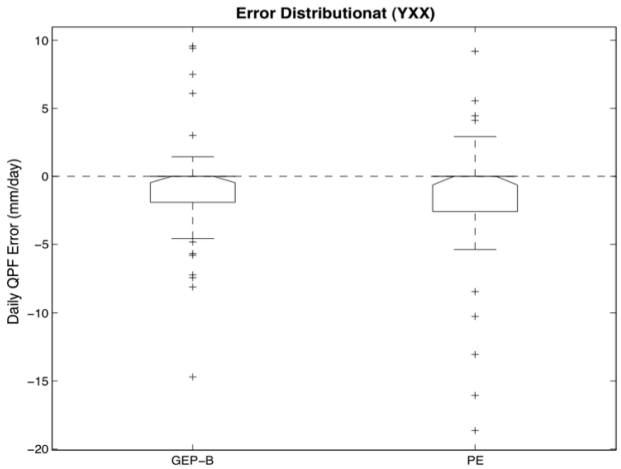


(a)





(b)



(c)

Figure F.4: Same as Figure F.1, but for Abbotsford Airport (YXX).

Appendix G

Functions Available to GEP for Electric Load Forecasting Symbolic Regression

For Chapter 3, this is a list of our manually chosen functions out of 279 built- in functions in the GEP software package from which GEP randomly draws during creation for the first generation and for mutation later on. Algorithms are filled with the weighted number of copies of each function listed below. "Arity" is the number of arguments for each function, which is important for determining the tail size of each gene. (x, y, z) are dummy arguments used by the function in the order listed.

Function	Symbol	Weight	Arity
Addition	+	4	2
Subtraction	−	4	2
Multiplication	⋆	4	2
Division	/	1	2
Square root	Sqrt	1	1
Exponential	Exp	1	1
Natural logarithm	Ln	1	1
x to the power of 2	X2	1	1
x to the power of 3	X3	1	1
Logistic(x)	Logi	1	1
Logistic(x,y,z)	Logi3	1	3
Cube root	3Rt	1	1
Sine	Sin	1	1
Cosine	Cos	1	1
Arctangent	Atan	1	1
if x < 0 OR y < 0, then 1, else 0	OR1	1	2
if x < 0 AND y < 0, then 1, else 0	AND1	1	2
if x <y, then 1, else 0	LT2B	1	2
if x <= y, then 1, else 0	GOE2B	1	2

Table G.1: The list of our manually chosen functions out of 279 built-in functions in GEP software package.

Appendix H

Sample of GEP Algorithm for Electric Load Load Forecasting

Sample of MATLAB algorithm used in Chapter 3 for the C2 key point created as output from GEP evolution. The weather-station identifiers are: YVR = Vancouver, YKA = Kamloops, YXS = Prince George, YYJ = Victoria.

```
%-------------------------------------------------------------------
% Code generated by GeneXproTools 4.0 on 20/09/2010 3:23:13 PM
% Training Samples:  684
% Testing Samples:   172
% Fitness Function:  RRSE
% Training Fitness:  756.157287868934
% Training R-square: 0.896448044971958
% Testing Fitness:   721.462700091804
% Testing R-square:  0.851529201562342
%-------------------------------------------------------------------

function result = gepModelC2_remainder(d)


G1C0 = 2.686768;
G1C1 = -5.535309;
G2C0 = 3.589996;
G2C1 = 9.988556;
G3C0 = 1.763703;
G3C1 = 8.902832;
G4C0 = -6.81781;
G4C1 = -5.301666;
G5C0 = -8.22055;
G5C1 = -9.96521;



C1_remainder = 1;
D = 2;
MM = 3;
YKA_PC2 = 4;
YKA_PPC2 = 5;
```

```
YKA_Tave_7_11 = 6;
YKA_TC2 = 7;
YKA_TdC2 = 8;
YKA_Tmax = 9;
YKA_Tmin = 10;
YKA_WSmax = 11;
YKA_WSmaxC2 = 12;
YVR_PC2 = 13;
YVR_PPC2 = 14;
YVR_Tave_7_11 = 15;
YVR_TC2 = 16;
YVR_TdC2 = 17;
YVR_Tmax = 18;
YVR_Tmin = 19;
YVR_WSmax = 20;
YVR_WSmaxC2 = 21;
YXS_PC2 = 22;
YXS_PPC2 = 23;
YXS_Tave_7_11 = 24;
YXS_TC2 = 25;
YXS_TdC2 = 26;
YXS_Tmax = 27;
YXS_Tmin = 28;
YXS_WSmax = 29;
YXS_WSmaxC2 = 30;
YYJ_PC2 = 31;
YYJ_PPC2 = 32;
YYJ_Tave_7_11 = 33;
YYJ_TC2 = 34;
YYJ_TdC2 = 35;
YYJ_Tmax = 36;
YYJ_Tmin = 37;
YYJ_WSmax = 38;
YYJ_WSmaxC2 = 39;


varTemp = 0.0;

varTemp = ((d(YXS_WSmax)+((d(C1_remainder)-d(YYJ_TC2))-(d(MM)*G1C0)))+gepGT2E(d(YYJ_PPC2),d(YKA_Tmin)));
varTemp = varTemp + (d(YXS_WSmaxC2)-((d(YYJ_Tmax)*(d(YXS_WSmax)-(d(YXS_Tmin)-G2C1)))/(d(D)^3)));
varTemp = varTemp + gepGT2E(gepGT2E(gepGT2E(log(d(D)),(G3C1^2)),d(YKA_WSmax)),(d(YYJ_Tmax)+d(YVR_WSmax)));
varTemp = varTemp + gepGT2E((d(YXS_WSmax)-gepOR1(gepGT2E(sin(d(YVR_WSmaxC2)),gepLT2A(d(YVR_Tmin),
d(YVR_PC2))),d(YYJ_WSmaxC2))),d(YYJ_Tmin));
varTemp = varTemp + gepGT2E(G5C1,((sin((atan(d(YXS_WSmaxC2))*d(D)))-d(D))^3));

result = varTemp;


function result = gepOR1(x, y)
if ((x < 0) | (y < 0)),
```

```
        result = 1;
else
        result = 0;
end

function result = gepLT2A(x, y)
if (x < y),
        result = x;
else
        result = y;
end

function result = gepGT2E(x, y)
if (x > y),
        result = (x+y);
else
        result = (x*y);
end
```

Appendix I

List of Predictors for Electric-load Key Point Residuals

List of predictors for the C1 residual in Chapter 3:

C1 residual from the day before

C4 residual from the day before

Code of the day (1=Sunday, ..., 7= Saturday, 9= Holiday)

Code of month (1= January, ..., 12 = December)

And weather variables at four cities (Vancouver, Victoria, Kamloops and Prince George) as follows:

Accumulated precipitation within the C1 time window (see Figure 3.1)

Accumulated precipitation code within the C1 time window

Temperature at the time that C1 occurs

Dew-point temperature average within the C1 time window

Maximum temperature during the day before

Minimum temperature within the C1 time window

Maximum wind speed during the current day

List of predictors for the C2 residual

C1 residual from the current day

Code of the day

Code of month

And weather variable at four cities (Vancouver, Victoria, Kamloops and Prince George) as follows:

Accumulated precipitation within the C2 time window

Accumulated precipitation code within the C2 window

Temperature at the time that C2 occurs

Average temperature within the C2 time window

Dew-point temperature at the time that C2 occurs

Maximum temperature during current day

Minimum temperature during the current day

Maximum wind speed during the current day

Maximum wind speed within the C2 time window

List of predictors for the C3 residual

C2 residual from the current day

Code of the day

Code of month

And weather variable at four cities (Vancouver, Victoria, Kamloops and Prince George) as follows:

Accumulated precipitation within the C3 time window

Accumulated precipitation code within the C3 time window

Temperature at the time that C3 occurs

Average temperature within the C3 time window

Dew-point temperature at the time that C3 occurs

Maximum temperature during current day

Minimum temperature during current day

Maximum wind speed during current day

Maximum wind speed within the C3 time window

List of predictors for the C4 residual

C3 residual from the current day

Code of the day

Code of month

And weather variable at four cities (Vancouver, Victoria, Kamloops and Prince George) as follows:

Accumulated precipitation within the C4 time window

Accumulated precipitation code within the C4 time window

Temperature at the time that C4 occurs

Average temperature within the C4 time window

Dew-point temperature at the time that C4 occurs

Maximum temperature during current day

Minimum temperature during current day

Maximum wind speed during current day

Maximum wind speed within the C4 time window

Appendix J

Illustration of a GEP Algorithm for One-hour-ahead Electric Load Forecasting

For Chapter 4, GEP devised the following algorithm for a single-stage, one-hour-ahead, electric load forecast for weekdays, based on the long training data set. The total stage-one load is the sum of the loads from separate genes: $L(t+1) = g1 + g2 + g3 + g4$, with

- $g1 = L + M - T - H - H^2 - 1.474$
- $g2 = L^{1/3}.[0.749 + \sin(H)]/\cos(6.071 + H)$
- $g3 = \ln(L).H.\sin(-7.319 - 6.816H)$
- $g4 = 104.71.\ln(H - 0.9806)$

where the first gene is the dominant gene in this case. The trigonometric functions treat their arguments as if the units were radians. All the input variables (M = month, T = temperature °C, and H = hour) are for the valid forecast time $(t+1)$, except L which represents the previous-hour load $L(t)$. The data points for GEP_1 in Figure 4.2 were calculated using this algorithm. Thus, even though GEP was allowed to use up to 5 genes, it used only 4 for this best solution. Also, GEP's solution did not include the previous-hour temperature $T(t)$ nor the day-of-week code $D(t+1)$, even though they were provided as input. It used only the following operators $(+, -, *, /, ()^2, ()^{1/3}$,exp, ln, sin, cos), and did not need the other operators provided for this weekday, stage-1 fit. Note that for weekends and holidays, GEP devised completely different stage-1 algorithms that look nothing like the equations above.

Appendix K

Sample Results

While Eq. 5.15 - Eq. 5.18 look complicated, they are simple to program or to enter in a spreadsheet. To allow readers to check their own code, sample calculations are given here.

Given an air parcel at its LCL of $P = 80$ kPa (i.e., 800 hPa) with temperature of $T = 9°C$, identify which saturated pseudoadiabat goes through this point. Eq. 5.15b - Eq. 5.15g give: $g_1 = -1.26$, $g_2 = 53.24$, $g_3 = 0.58$, $g_4 = -8.84$, $g_5 = -25.99$, and $g_6 = 0.15$. Summing these in Eq. 5.15a gives $\theta_w = 17.9°C$, which is very close to the skew-T and tephigram values of slightly less than 18°C.

For cold θ_w air, given a saturated air parcel that follows the $\theta_w = -10°C$ saturated pseudoadiabat up to an altitude where the pressure is $P = 70$ kPa (i.e., 700 hPa), find its air temperature. Eq. 5.16b - Eq. 5.16g give: $g_1 = -22.10$, $g_2 = 67.99$, $g_3 = 0.73$, $g_4 = -68.04$, $g_5 = 0.27$, and $g_6 = -10.98$. Summing these in Eq. 5.16a gives $T = -32.1°C$, which is very close to the skew-T and tephigram values of $-32.2°C$.

For medium θ_w air, given a saturated air parcel that follows the $\theta_w = 16°C$ saturated pseudoadiabat up to an altitude where the pressure is $P = 40$ kPa (i.e., 400 hPa), find its air temperature. Eq. 5.17b - Eq. 5.17h give: $g_1 = -10.5$, $g_2 = 16.4$, $g_3 = 3.4$, $g_4 = 11.9$, $g_5 = 39.7$, $g_6 = 3.5$, and $g_7 = -93.6$. Summing these in Eq. 5.17a gives $T = -29.3°C$, which is close to the skew-T value of -28.6 and the tephigram value of $-28.2°C$.

For hot θ_w air, given a saturated air parcel that follows the $\theta_w = 28°C$ saturated pseudoadiabat up to an altitude where the pressure is $P = 25$ kPa (i.e., 250 hPa), find its air temperature. Eq. 5.18b - Eq. 5.18h give: $g_1 = 0.9$, $g_2 = -33.9$, $g_3 = -18.2$, $g_4 = 6.8$, $g_5 = 30.2$, $g_6 = -13.8$, and $g_7 = 0.9$. Summing these in Eq. 5.18a gives $T = -27.2°C$, which is close to the skew-T value of -27.1 and the tephigram value of $-26.5°C$.